

Premios para los lectores • Estrategia • C y Ensamblador • Aventura • DIV 2 • Arcade
CONCURSO JUEGOS DIV
DIV manía
995 ptas.
PORTUGAL 990 ESC (CONT) 5,98 €
www.prensatecnica.com
Año 1 • Número 3
•DIV 2
Se consolida la segunda edición de la herramienta
•Webs DIV
La red se rinde a la segunda versión del entorno
•DirectX
Nociones básicas de las librerías de Microsoft
•Photoshop
Curso de tratamiento de imágenes
•DIV Interno
Lo más íntimo de DIV
•Programación
Análisis específico de cada género
ESTE MES nuevas SECCIONES
Las herramientas de los programadores
Turbo C++ Suite
C++Builder 4
Coming Soon!
Prens@ Técnic@
8 413042 087533
00003

# Tu PC siempre a punto con CD DRIVER

**Tarjetas gráficas, impresoras, BIOS, unidades Zip, discos duros, placas base, memorias, escáneres, monitores, digitalizadoras, modems, coprocesadores , joysticks, tarjetas de sonido...**

## ¿Qué contiene CD Driver?

**Más de 1.500 drivers**

**Alrededor de 900 horas de búsqueda en Internet**

**Con este número te ahorrarás más de 100.000 ptas. en gastos de Internet y teléfono**



## ¿Qué es CD Driver?

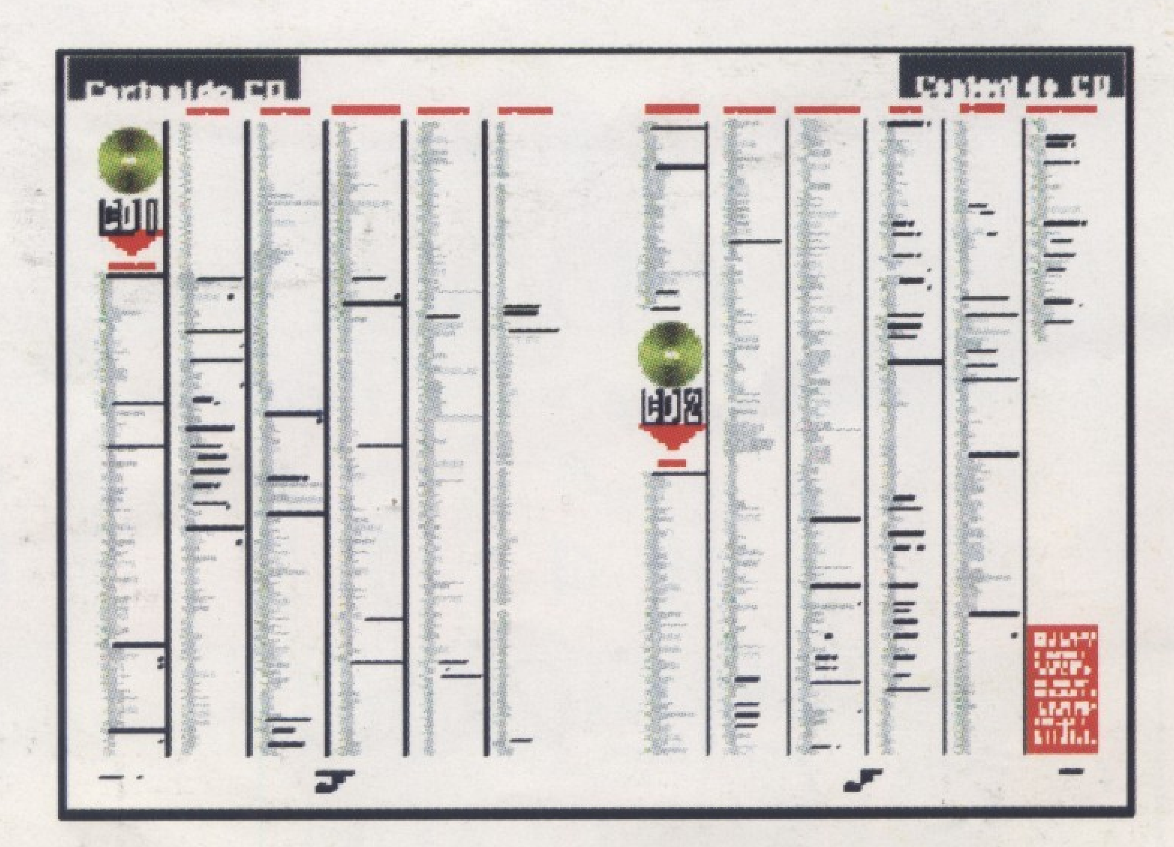
***CD DRIVER es la revista que irrumpe en el mercado para hacer más fácil la vida del aficionado a la informática.***

***Todo el mundo ha experimentado la dificultad que supone conseguir un determinado driver o, simplemente, mantenerlos juntos y ordenados.***

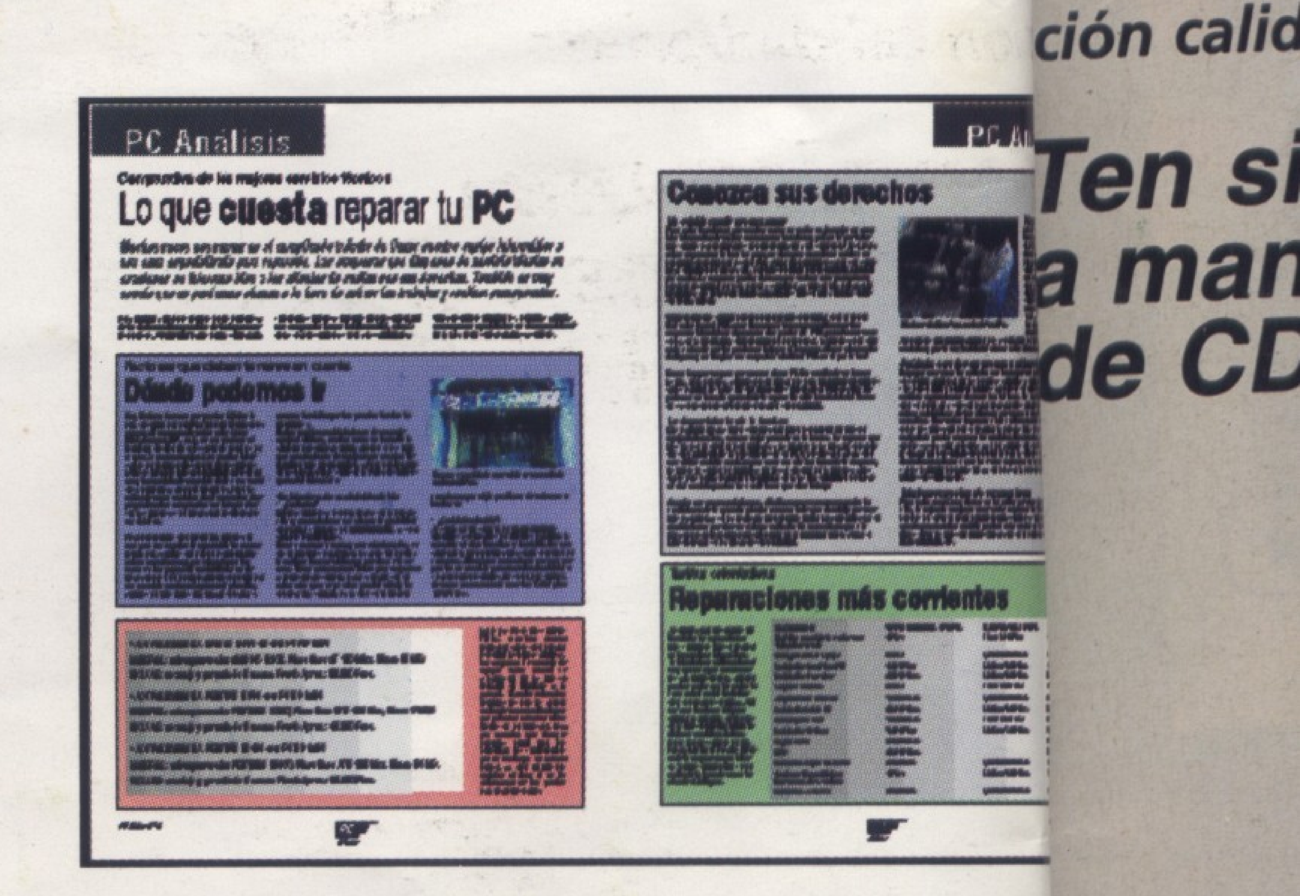
***CD DRIVER contiene en sus 2 CD ROMS el compendio de todos los drivers, modelos y sistemas, actualizaciones y viejas versiones de las marcas más distribuidas por todos los fabricantes de Hardware, para tener tu PC siempre a punto.***

## ¿Sabes resolver el Puzzle de la Informática?

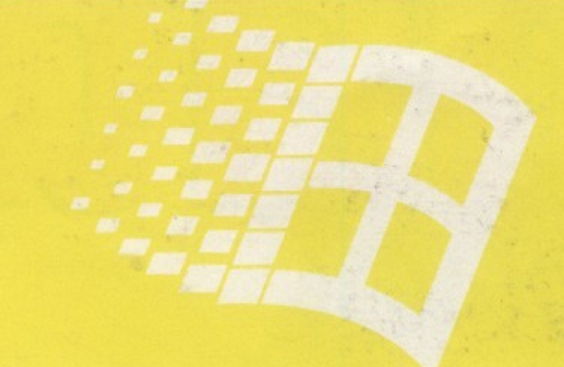
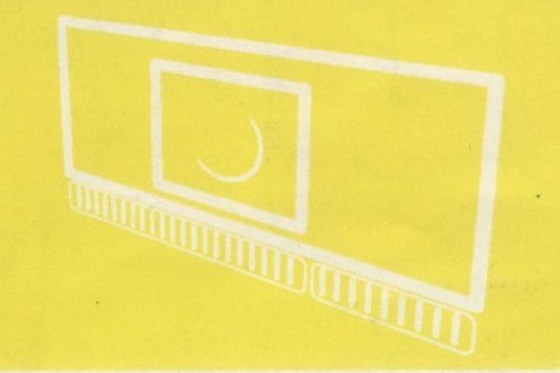
Soluciones para las averías más comunes.



Directorios de todos los drivers de los CD-Roms.




Analizamos los mejores servicios técnicos para P

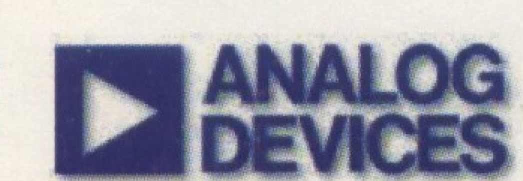





TODAS LAS MARCAS SON PROPIEDAD DE SUS RESPECTIVOS FABRICANTES.





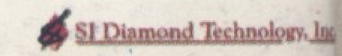












# DIVmanía



## ¿Amas la programación de juegos...? Léete estas líneas

*Hemos cruzado ya el ecuador de un verano abrasador, pero nosotros seguimos al pie del cañón, con la firme intención de consolidar nuestro lugar entre todos los usuarios interesados en la programación de juegos. Ni los más fuertes calores del año pueden disminuir la pasión que los juegos despiertan en todos los que se acercan a ellos. En este tercer número estamos asentando una revista que cada vez tiene una mayor aceptación entre todos vosotros, quizá por el fuerte tirón que tiene DIV 2 .*

*En cualquier caso, nos estamos afianzando y para ello hemos empezado a ampliar nuestras miras con temas cómo, el curso de DirectX. Es imprescindible dominar completamente estas librerías para poder enfrentarse hoy en día a la programación de juegos. Nuestro reportaje central gira, en cualquier caso, alrededor de las herramientas que necesita un programador o, mejor, un grupo de programación, para poder realizar un videojuego de calidad profesional. Desde luego, DIV sigue siendo un estupendo entorno de desarrollo del que partir para realizar un juego, pero en la mayor parte de compañías desarrolladoras se parte de los grandes programas. Así pues, aquellos que deseen acabar en un grupo de programación profesional tendrán que ponerse al día en los programas que comentamos.*

*La respuesta de nuestros lectores está siendo excelente. Cada vez nos llegan más juegos programados por ellos, y su nivel es más alto cuando más expertos son sus realizadores. Hemos de decir, también, que ya han llegado los primeros títulos programados con la segunda versión de DIV que, a pesar de las distintas opiniones al respecto, ya ha empezado a calar hondo entre todos los aficionados.*



Secretos de los juegos





Imprescindibles librerías

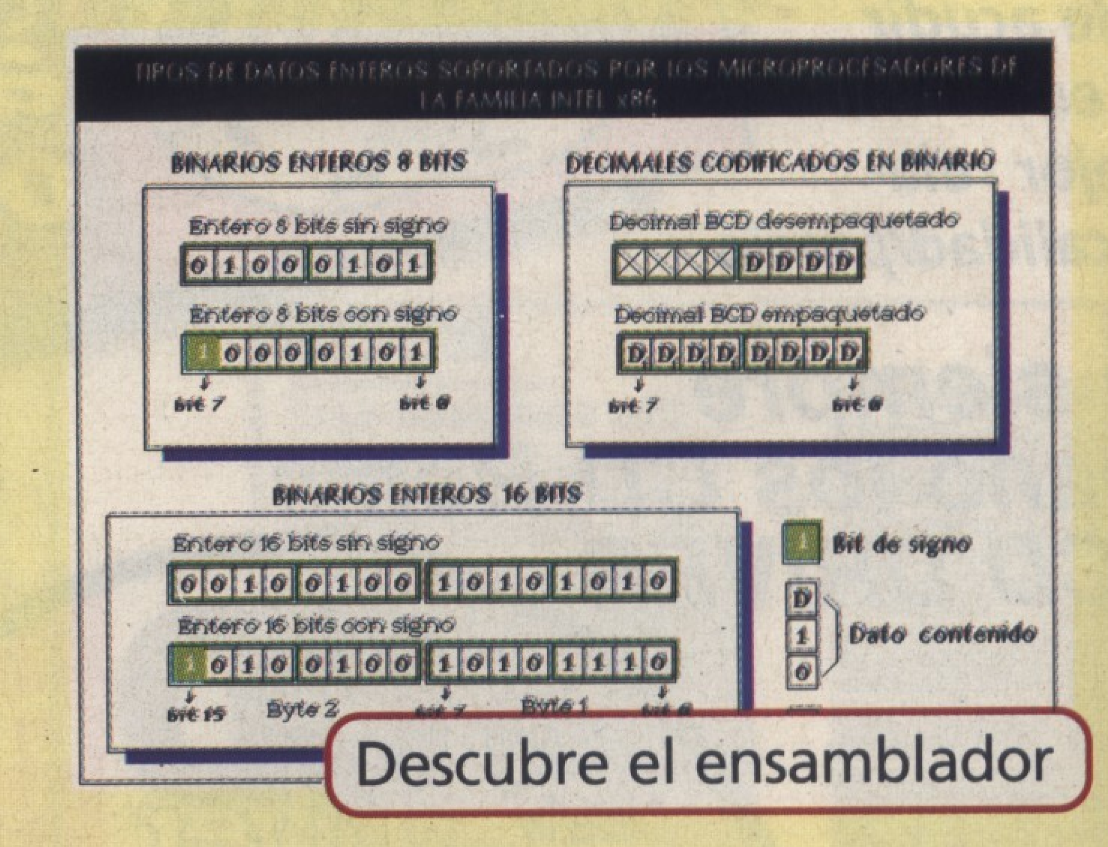


## DIV Developer










Descubre el ensamblador


**Director:** Mario Luis
mluis@prensatecnica.com
**Coordinador Técnico:**
Rafael María Claudín
divmania@prensatecnica.com
**Colaboradores:** Antonio Marchal, Pablo Trinidad, José M. Sevillano, Sergio Cánovas, Miguel Barroso, David Martínez, Emilio Llanos.
**Edición:** Óscar Condés y Alfredo del Barrio
**Dirección de Arte:** Francisco Calero

**Maquetación:** Manuel J. Montes, José A. Gil, Antonio Barbero, José Mª Gil, Ana Isabel Madero y Maribel Grande
**Portada:** Francisco A. Anguís
**Publicidad:** Marisa Fernández, marisa@prensatecnica.com
Sonia Glez.-Villamil, Noelia Menéndez y Emerio Arena
**Supervisión CD-Rom:**
Alvaro García y Noé Soriano

**Servicio Técnico CD-Rom:**
Eugenio García
Horario de atención:
tardes 16:00 - 18:00 h
E-mail: stecnico@prensatecnica.com
**Secretaria de Redacción:**
Elena Fernández
**Departamento de Suscripciones:**
Sandra Fernández y Noemí Iscart
suscripciones@prensatecnica.com
**Departamento de Administración:**
Juan López, Juan Ignacio Domínguez

**Departamento Comercial:**
Marcelino Ormeño

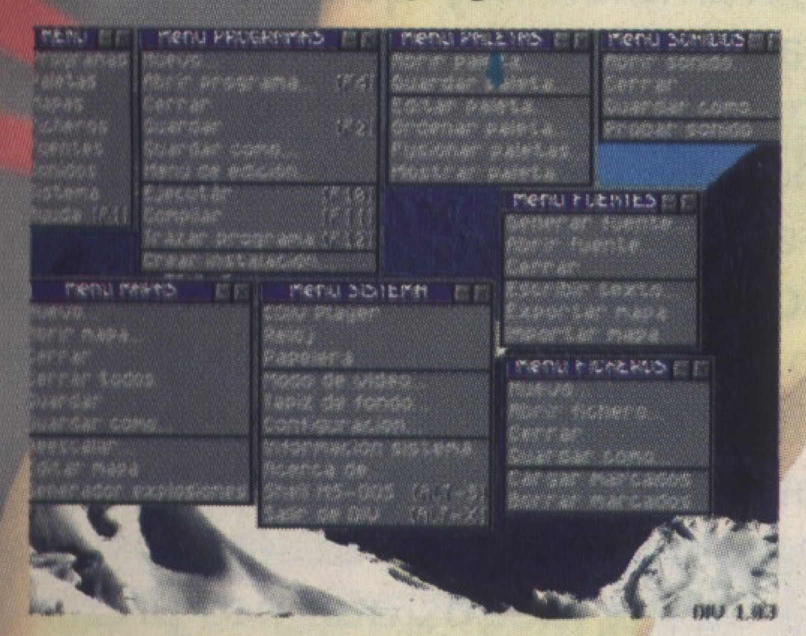
**REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN**
c/ Alfonso Gómez 42. Nave 1.1.2
Madrid 28037. España
Tfno: (91) 304. 06. 22
Fax: (91) 304. 17. 97
Si llama desde fuera de España, marcar (+34)
E-mail: ntactual@prensatecnica.com
http://www.prensatecnica.com

# Sumario





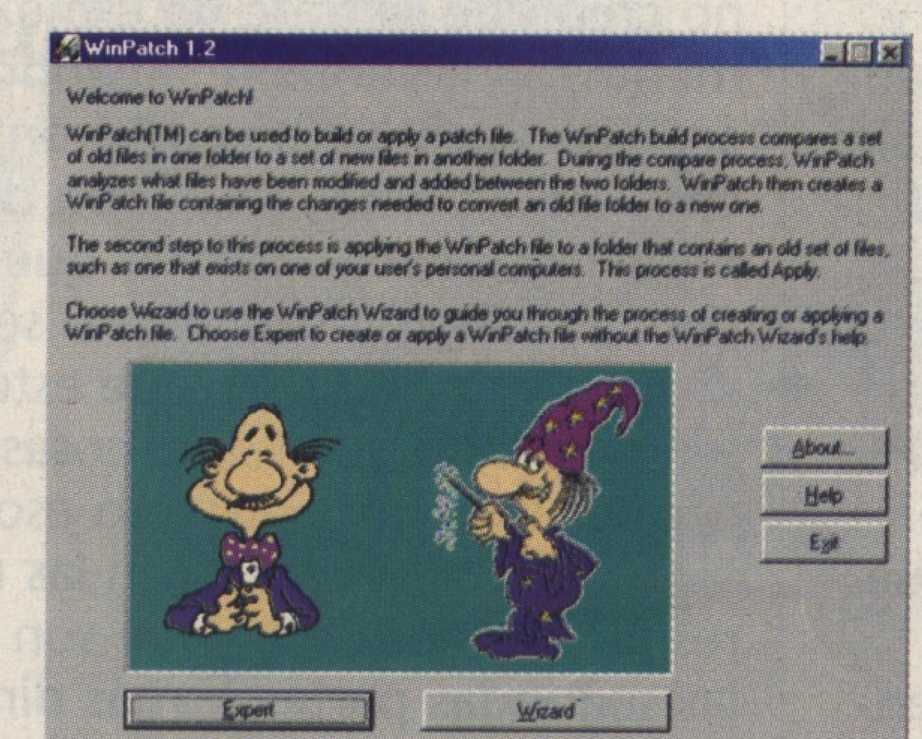











## Programas del lector





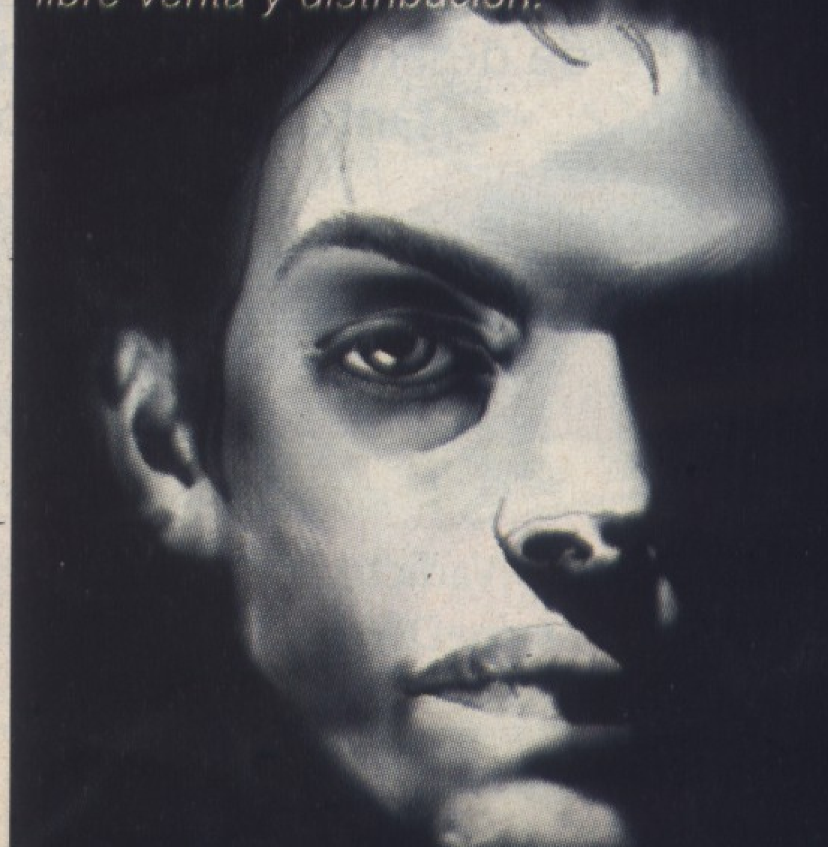



**PREMIAMOS LOS MEJORES JUEGOS DE LOS LECTORES**

*DIV ha creado toda una escuela dentro del mundo de la programación. Todos los que se han acercado a la herramienta han sido capaces de programar. Por ello realizamos un concurso entre los lectores que hayan realizado juegos con DIV. Os ofrecemos un primer premio de 25.000 pesetas y dos accésit de 20.000 pesetas.*

## ¿QUÉ ES DIV GAMES STUDIO?

*DIV Games Studio es una herramienta de programación que facilita en gran manera nuestra inmersión en el software de entretenimiento. Es el primer entorno profesional que permite realizar videojuegos con fines comerciales sin necesidad de un pago adicional. Con el carné de desarrollador incluido se permite el desarrollo de cualquier número de videojuegos y su libre venta y distribución.*




Horario de atención al público: de 09:00 a 19:00 h ininterrumpidamente

**EDITA: PRENSA TÉCNICA**
**Director General:** Mario Luis
**Director Editorial:** Eduardo Toribio
**Director Técnico:** Fernando Escudero
**Director de Producción:** Carlos Peropadre
**Director Financiero:** Felipe Hernandez
**Directora Publicidad:** Marisa Fernández
**Director Comercial:** Esteban Martinez
**Fotomecánica:** Grafoprint
**Impresión:** Printerman
**Duplicación del CD-Rom:** M.P. O., Servicios Ibéricos, Grupo Cóndor

**Distribución:** SGEL. Avda Valdelaparra, 29 Alcobendas. Madrid





**Depósito legal:** M-42077-1998
AÑO 1 • NÚMERO 3

# El lector lúdico

## Sólo para aficionados a los juegos

**DIV 2 ya tiene unos cuantos meses de vida, y hasta el momento está obteniendo una excelente respuesta por parte de todos los aficionados a este entorno de programación. Así pues, a pesar de estar en medio de un caluroso verano, hemos decidido salir a la calle para llegar hasta vosotros.**

No tenemos en esta ocasión una colección de poemas con la que deleitar a nuestros lectores y hacernos eco de lo que sucede en la web y en los círculos de DIV. Sin embargo, sí queremos reflejar el interés que la segunda edición de esta herramienta de trabajo ha despertado en distintos medios, muchos de ellos no especializados, lo que indica que DIV está empezando a trascender algo más allá.

Ha habido, de todas maneras, todo tipo de opiniones respecto a DIV 2. Hay quien piensa que no añade nada respecto a la primera edición. Otros dicen que únicamente corrige errores; y si es así, ¿cuántos errores tenía DIV? También están, desde luego, los que defienden este entorno de desarrollo a uñas y dientes. Ni tanto ni tan calvo, desde luego.

Lo cierto es que esta segunda parte ha dado menos de lo que se esperaba. Y, sin embargo, sigue mereciendo la pena. Las opciones 3D, que muchos esperaban como agua de mayo, resultó que al final no eran tan completas como se esperaba. Así, parece que crear nuestro propio Doom no va a ser tan fácil como esperábamos en un principio. Y, sin embargo, no podemos negar que, en ese sentido, la puerta está abierta. De hecho, uno de los cursillos que os ofrecemos nos enseñará en breve a realizar juegos tipoY *¡Tomb Raider!* Pero no adelantemos acontecimientos; además, el problema añadido de conseguir una chica como Lara Croft tendréis que resolverlo vosotros.

Respecto a lo de la corrección de errores, bueno, lo cierto es que no creo que se trate de eso exactamente. Sencillamente, cuando uno acaba un producto muy pronto empieza a pensar en el tipo de modificaciones y mejoras que le puede realizar. Así pues, no se trata de que hubiese cosas que estuviesen mal, sino que había cosas susceptibles de mejora. ¡Pero eso es algo que sucede hasta en las mejores familias lúdicas! ¿Qué son si no las segundas partes y, más directamente, las actualizaciones de juegos que ya tenían una alta calidad en su primera aparición?

Y, si observamos con cierto detenimiento DIV 2, nos damos cuenta de que el perfeccionamiento en todos sus aspectos es considerable. Tizo, por ejemplo, uno de los más fieles divmaníacos, recomendaba sin duda la nueva edición de la herramienta, y él la conoce, podemos decirlo, bien a fondo.

Por lo demás, tenemos que decir que las nuevas aportaciones del segundo número de DIV Manía fueron bien recibidas por parte de los lectores, a juzgar por las notas que nos han ido enviando, de modo que hemos decidido seguir en la línea y ampliar las miras de la programación de juegos mucho más allá de los límites de DIV.



También hay que decir que la mayor parte de cartas recibidas por nuestros lectores eran en realidad pequeños programas que se presentan al concurso que ofrecemos cada mes, motivo por el cual no hemos realizado un tête a tête entre los lectores y nosotros. Los comentarios que nos han hecho han sido más bien accesorios dentro de los juegos que nos presentan. Sí nos han hecho notar algunos lectores que faltaban los programas necesarios en el CD-Rom para seguir convenientemente algunos de los pequeños cursillos que realizamos. Pues bien, el error queda subsanado con la inclusión de los pequeños programitas que sirven para ilustrar los artículos.

(*) DIVadicto, DIVo, DIVa: Dícese de aquel aficionado a la programación de videojuegos que se encuentra fascinado por DIV. ImpeDIVmentos: En lenguaje de DIVos, se trata de los problemas a los que se enfrenta un desarrollador que use DIV.




- Prensa Técnica te ofrece los últimos avances y novedades del mundo de la informática a través de sus publicaciones.
- Internet, Linux, Diseño digital, Programación, Juegos... una oferta variadísima que cubre todo lo que necesitas para estar al día.
- Tenemos revistas para todos los públicos, ya seas principiante o avanzado, Prensa Técnica tiene la solución a tus problemas.
- Súbete al tren del saber y no te pierdas las últimas noticias, informaciones y cursos.




**LA REVISTA QUE TE DA MÁS**
MÁS PC, la revista informática para todos los públicos, con toda la información y actualidad en hardware, software, Internet, diseño, Linux, programación, videojuegos, multimedia, etc.
Pc
*Incluye CD-Rom y libro técnico*

**¡Más de 350.000 lectores cada mes!**




**TU ORDENADOR AL DÍA**
CD DRIVER es una revista imprescindible para el mantenimiento de tu PC. Con ella, el usuario informático tendrá a mano todos los drivers del mercado y estupendos artículos sobre la utilización e instalación de los componentes del PC.
Pc
*Bimestral* *Incluye 2 CD-Roms*

**TU GUÍA PARA LA RED**
INTERNET ONLINE se introduce en los recovecos de la Red mostrándole información rigurosa sobre aspectos técnicos, análisis de webs y herramientas. Incluye CD-Rom con navegadores, utilidades de correo, chat, etc.
Pc • Mac
*Incluye CD-Rom*

**LA MÁS VENDIDA DE EUROPA**
ELECTRÓNICA PRÁCTICA ACTUAL es la edición en castellano de la revista de electrónica más vendida de Europa. Contenidos prácticos de electrónica e informática con noticias, Internet y los montajes más ingeniosos.
Pc
*Incluye CD-Rom*

**LA MEJOR RECOPILACIÓN**
MODELOS 3D es la revista que te proporciona todos los modelos y texturas que necesitas sin tener que perder el tiempo buscándolas. Incluye modelos, texturas y demos de los programas 3D más utilizados.
Pc • Mac
*Bimestral* *Incluye CD-Rom*




**CREAR ESTÁ EN TUS MANOS**
3D WORLD está especializada en infografía y en general las 3D. Con la última actualidad en diseño gráfico, reportajes, técnicas, trucos y tutoriales de los programas de diseño y 3D más utilizados en el sector profesional.
Pc • Mac
*Incluye CD-Rom*

**LA NUEVA ERA DE LA FOTOGRAFÍA Y EL ARTE**
FOTO ACTUAL Y ARTE DIGITAL, revista para profesionales y aficionados al diseño, maquetación y retoque fotográfico. La mejor forma de conocer toda la teoría y la práctica sobre las técnicas más utilizadas del momento.
Pc • Mac
*Incluye CD-Rom*

**JUGANDO DURO**
GAME OVER analiza los juegos de ordenador desde el punto de vista de los propios creadores. Toda la información técnica además de un análisis riguroso de las últimas novedades del mercado.
Pc
*Incluye CD-Rom*

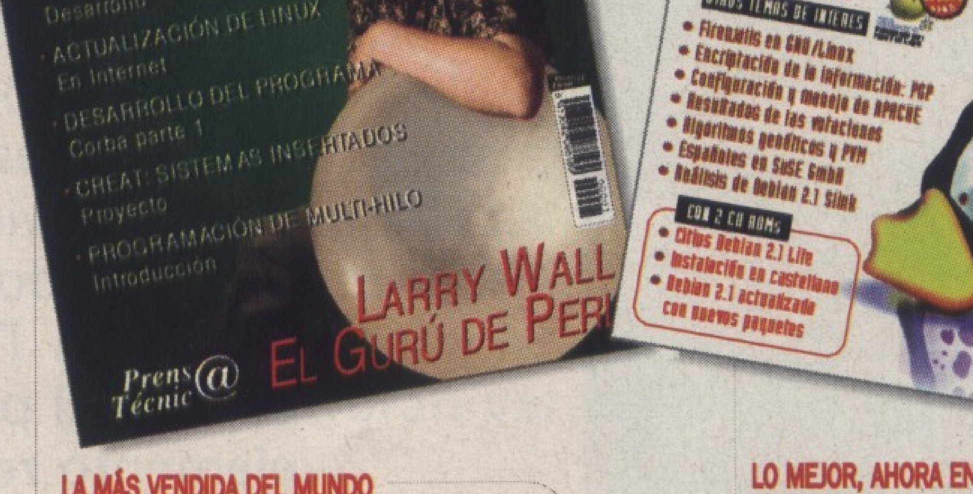
**NUNCA DEJES DE JUGAR**
ZONA PSX STATION encuentra una nueva dimensión para tu Playstation con una revista llena de originales secciones, objetiva y con un diseño que da a las imágenes la importancia que se merecen.
Incluye suplemento Xtreme PSX

**HAZ TUS PROPIOS VIDEOJUEGOS**
DIV MANÍA es la primera revista dedicada a aprender a programar videojuegos, abarcando todos los aspectos del desarrollo. Incluye CD-Rom con tres juegos programados por los lectores y demos de juegos profesionales.
Pc
*Bimestral* *Incluye CD-Rom*










**POR Y PARA PROGRAMADORES**
PROGRAMACIÓN ACTUAL te pone al día del mundo del desarrollo gracias a sus secciones principales dedicadas a la programación gráfica, Internet y sus lenguajes, desarrollo empresarial y nuevas tecnologías.
Pc
*Incluye CD-Rom*

**LO ÚLTIMO EN TECNOLOGÍA**
WINDOWS NT ACTUAL está destinada a profesionales del mundo NT. El modo más fácil para estar al día y conocer el entorno NT así como sus aplicaciones.
Pc
*Incluye CD-Rom*

**LA MÁS VENDIDA DEL MUNDO**
LINUX JOURNAL es la edición en nuestro país de la publicación más prestigiosa del mundo GNU/Linux. Entrevistas, actualidad y buenos artículos se dan cita en una auténtica "BIBLIA" sobre este sistema operativo.

**LO MEJOR, AHORA EN CASTELLANO**
LINUX ACTUAL es la primera revista en castellano dedicada al GNU/Linux: el sistema operativo de moda. Incluye artículos dedicados a todas sus áreas y un CD-Rom con las mejores distribuciones y novedades del momento.
Pc
*Bimestral* *Incluye CD-Rom*

**PENSADA PARA PRINCIPIANTES**
SÓLO LINUX es la mejor revista en castellano para el usuario principiante en el mundo GNU/Linux. En ella encuentra toda la información en forma de artículos de nivel básico. Incluye un CD-Rom con la distribución más fácil de instalar del momento.
Pc
*Bimestral* *Incluye CD-Rom*

Edita **PRENSA TÉCNICA** de libros y publicaciones • Alfonso Gómez 42, Nave 1-1-2. • 28037 Madrid • Tf: 91 3.04.06.22 • Fax: 91 3.04.17.97

# Electro diseño interactivo y el VRML



Electro Diseño Interactivo es una división de Electro Criaturas Virtuales, S.L., una empresa cuyo enfoque se basa en el desarrollo de la realidad virtual con VRML (Virtual Reality Modeling Language), un campo de enormes posibilidades aún sin explotar, a pesar de que funciona perfectamente en equipos domésticos. La compañía está especializada en diseñar soluciones y productos destinados a la publicación en Internet y soportes de almacenamiento para la navegación off-line, principalmente productos multimedia basados en la interactividad.

Sus servicios abarcan las siguientes áreas: Creatividad (conceptualización de entornos virtuales, creación o adaptación de la imagen de marca al medio digital), Estructuración y Estrategias (elaboración de planes específicos de navegación, servicios virtuales y fidelización de usuarios) y Programación en VRML (además de lenguajes HTML, DHTML, FLASH, etc., según cada proyecto). Entre sus productos destacan: Realidad Virtual con VRML, Websites, Banners, CD-Rom y DVD-Rom, y otros relacionados con la imagen y el sonido.

Sin duda el aspecto más interesante de su trabajo es el que se refiere al VRML, el lenguaje utilizado para la descripción de mundos tridimensionales en Internet. Se trata de archivos de pequeño tamaño que se cargan rápidamente desde la red y permiten abrir ventanas a la tercera dimensión donde los usuarios cuentan con libertad para explorar escenarios virtuales, examinar e interactuar con objetos, etc. VRML se puede considerar como un lenguaje multimedia al integrar animación, audio y vídeo.

VRML es totalmente compatible con HTML, siendo capaz de realizar hipervínculos desde el mundo tridimensional hasta las páginas web (bidimensionales). Por otro lado, VRML no requiere de dispositivos externos especiales (los conocidos cascos y guantes digitales) para la navegación ya que basta con el ratón y el teclado. El único requerimiento especial es tener instalado en el navegador de Internet un plug-in que interprete el lenguaje. Estos plug-in pueden cargarse desde las páginas de los fabricantes de forma gratuita.

Las aplicaciones que Electro Diseño Interactivo realiza a partir del VRML abarcan diversos campos como el entretenimiento (videojuegos tridimensionales), la animación (Banners animados e interactivos, spots de productos, trailers tridimensionales de películas, series animadas), la simulación arquitectónica (representaciones tridimensionales que permiten visitar en tiempo real proyectos aún no construidos, incluyendo posibilidades como examinar objetos y encender o apagar luces), la educación, las presentaciones y las panorámicas (experiencias inmersivas en entornos de realidad virtual).

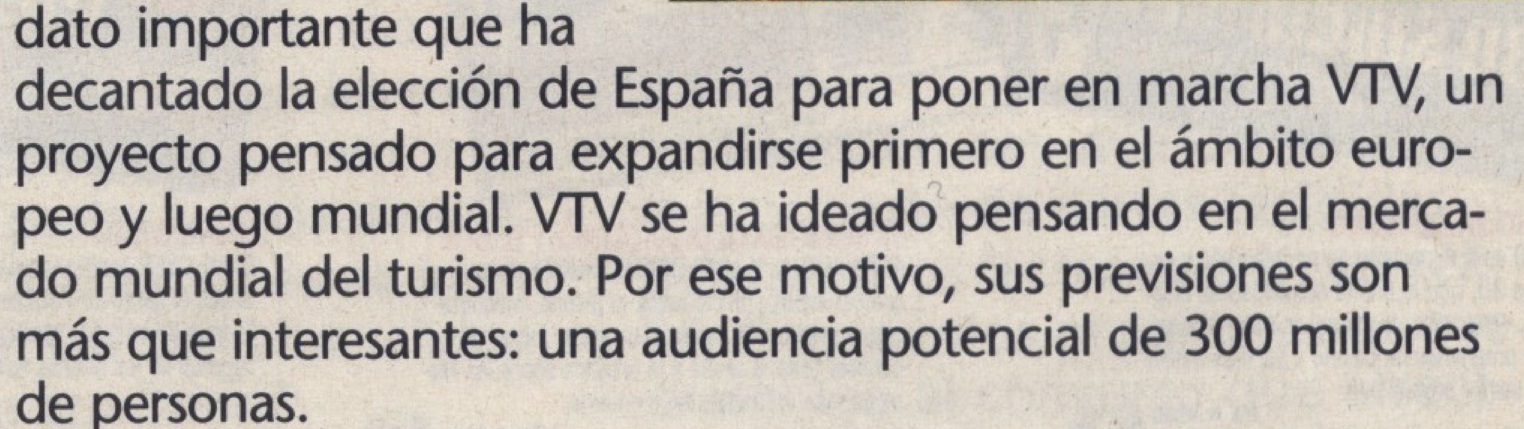
# VTV Canal Vacaciones




El pasado día 6 de Julio se presentó en Madrid VTV Canal Vacaciones, el primer canal de televisión interactivo y multimedia dedicado íntegramente al sector turístico.

Aunque ya emite regularmente desde Marzo pasado, el nuevo canal supone una nueva vía de comunicación entre profesionales del sector turístico y los espectadores, que ahora dispondrán de un canal de entretenimiento en el que, además, podrán encontrar información personalizada para sus viajes: infraestructuras hoteleras, servicios existentes, información cultural, histórica o natural sobre el destino, transportes y, finalmente, precios vigentes en el mercado facilitados por los operadores turísticos más cercanos al lugar de residencia del interesado.

Se trata de un proyecto original europeo en el que participan profesionales de los sectores turístico y audiovisual, cuya inversión inicial se cifra en 4,67 millones de euros. La posición privilegiada de nuestro país en el mercado del turismo mundial es un dato importante que ha



decantado la elección de España para poner en marcha VTV, un proyecto pensado para expandirse primero en el ámbito europeo y luego mundial. VTV se ha ideado pensando en el mercado mundial del turismo. Por ese motivo, sus previsiones son más que interesantes: una audiencia potencial de 300 millones de personas.



# Omikron cambia de nombre

Según hemos podido saber a través de diversas fuentes, Eidos ha decidido variar el título de su esperado juego de lucha y aventura, hasta ahora conocido como *Omikron: The Nomad Soul,* y que saltó a la palestra con el anuncio de que el mismísimo David Bowie iba a colaborar en él muy activamente. Eidos ha decidido suprimir la primera palabra y dejarlo simplemente en *The Nomad Soul.* Un portavoz de la compañía explicó el porqué: "el nuevo título centra la atención en el elemento reencarnación del juego, que es probablemente la parte más novedosa".

# GameLaunch 3D, Versión 1.5

Se trata de un software de configuración de juegos que permite a los usuarios customizar sus tarjetas 3D y los controles del juego. GameLaunch 3D permite optimizar las tarjetas Voodoo y RIVA TNT para juegos basados en los motores Quake de Id Software. La nueva versión añade soporte para Riva TNT y Voodoo 3, así como para varios nuevos juegos. La configuración *drag-and-drop* ha sido mejorada también. Las nuevas características incluyen soporte para Voodoo Banshee, Voodoo 3, Riva TNT, Riva TNT2, Riva TNT2 Ultra, Team Fortress Classic, Quake III Arena y un conversor de *script* para convertir antiguas configuraciones a juegos nuevos.

# Kazoo 2.0 de Light Works

LightWorks ha lanzado la versión 2 de Kazoo, su software para la creación de entornos interactivos 3D. El nuevo programa trae nuevas posibilidades para los desarrolladores de aplicaciones 3D, en el campo de la edición, retoque y composición de la imagen digital. Con Kazoo, los usuarios pueden agregar clips artísticos 3D como broche a sus fotos y también poner las fotos directamente en objetos y caracteres 3D. Los usuarios de Kazoo pueden elegir entre una amplia gama de modelos tridimensionales, combinar los modelos con sus fotos, variar el tamaño, rotar y hacer zoom sobre el modelo para conseguir el ángulo perfecto. Kazoo también permite iluminar y sombrear, además de una amplia variedad de funciones para pintar los objetos 3D, incluyendo lápiz, tinta, pintura al agua, caricatura, mosaico y carboncillo.

Kazoo 2.0 tiene cinco módulos: visualizador (posibilita al usuario de PC que pueda manipular objetos 3D e insertarlos después en documentos o presentaciones), codificador (convierte los modelos 3D en archivos de formato propio del programa), paleta de aplicaciones (componentes de software de alto nivel que permiten diversos usos), pack de estilos (colecciones adicionales de estilos artísticos) y, finalmente, el reproductor (que permite enviar objetos e imágenes a no-usuarios de Kazoo para su visionado interactivo.

# Merlin VR de Digital Inmersion

Digital Inmersion´s ha lanzado el software Merlin VR que permite modelar y animar en tiempo real usando Direct3D. Merlin VR permite configurar fácilmente los objetos animados o las trayectorias de cámara. Algunas de sus características: aceleración previa en tiempo real, escenas animadas totalmente texturadas a la mayor cantidad de frames y posibilidad de ser guardada como un archivo .AVI para su inspección o distribución.

La interfaz de ususario de Merlin VR ofrece gran variedad de aspectos: 3D primitives, cloning, multi-clone; escala uniforme y no-uniforme, funciones booleanas avanzadas, aplicaciones de modelado en twisting, escalaje y deformaciones, agrupación jerárquica, luces y sombras en tiempo real, coloreado de luces, niebla en tiempo real y múltiples opciones de manipulación de cámara. Merlin VR es una herramienta que puede utilizarse para crear gran variedad de webs y contenidos multimedia o que puede ser usado por diseñadores que quieren mostrar sus conceptos de diseño en 3D. Merlin VR requiere un Pentium 133MHz con 32MB de RAM y 200MB de espacio libre; además, se recomienda tener una tarjeta gráfica de 4 Megas o superior.

# Friendware




## Líder en distribución

**La empresa Friendware es una de las más importantes plataformas de distribución de software de entretenimiento españolas. La compañía distribuye sus productos tanto en el ámbito de nuestro país, sus juegos llegan a los principales puntos de venta de toda España, como en America del sur, donde está presente en todos los países latinos.**

Algunos de los títulos distribuidos por esta empresa se han situado entre los primeros puestos en las listas de venta, además de haber obtenido muy buenas críticas por parte de la prensa especializada. Por otro lado, Friendware se ha convertido en la primera empresa del sector que apuesta fuerte por el software español. El primer título producido bajo el ámbito de Friendware, *Speed Haste,* fue uno de los primeros productos para PC españoles que se vendió en todo el mundo.

De capital íntegramente español, desde su fundación la compañía siempre se ha caracterizado por ser pionera en lo que a presencia de sus productos en Internet se refiere, además de ser una de las empresas de videojuegos que más se preocupa del soporte a los jugadores. En su página Web, Friendware ofrece su catálogo de productos, fechas de lanzamiento, pantallas, demos y un foro de usuarios, y muchas otras informaciones y servicios.

Su catálogo de productos presenta programas tanto de productoras nacionales como grandes compañías multinacionales. A continuación, vamos a mostraros algunos de los juegos más destacados que Friendware nos ofrece en estos momentos.

### Rolland Garros 99

El juego de tenis oficial de uno de los torneos más aclamados del tenis mundial.

Con respecto a la versión de 1.998, esta renovada versión incluye tecnología 3D actualizada, jugabilidad altamente mejorada, nuevo sistema de menús, texturas modificadas e incorporación de la Enciclopedia de la historia del torneo desde 1.968, con 250 fotos y el historial de 168 jugadores.

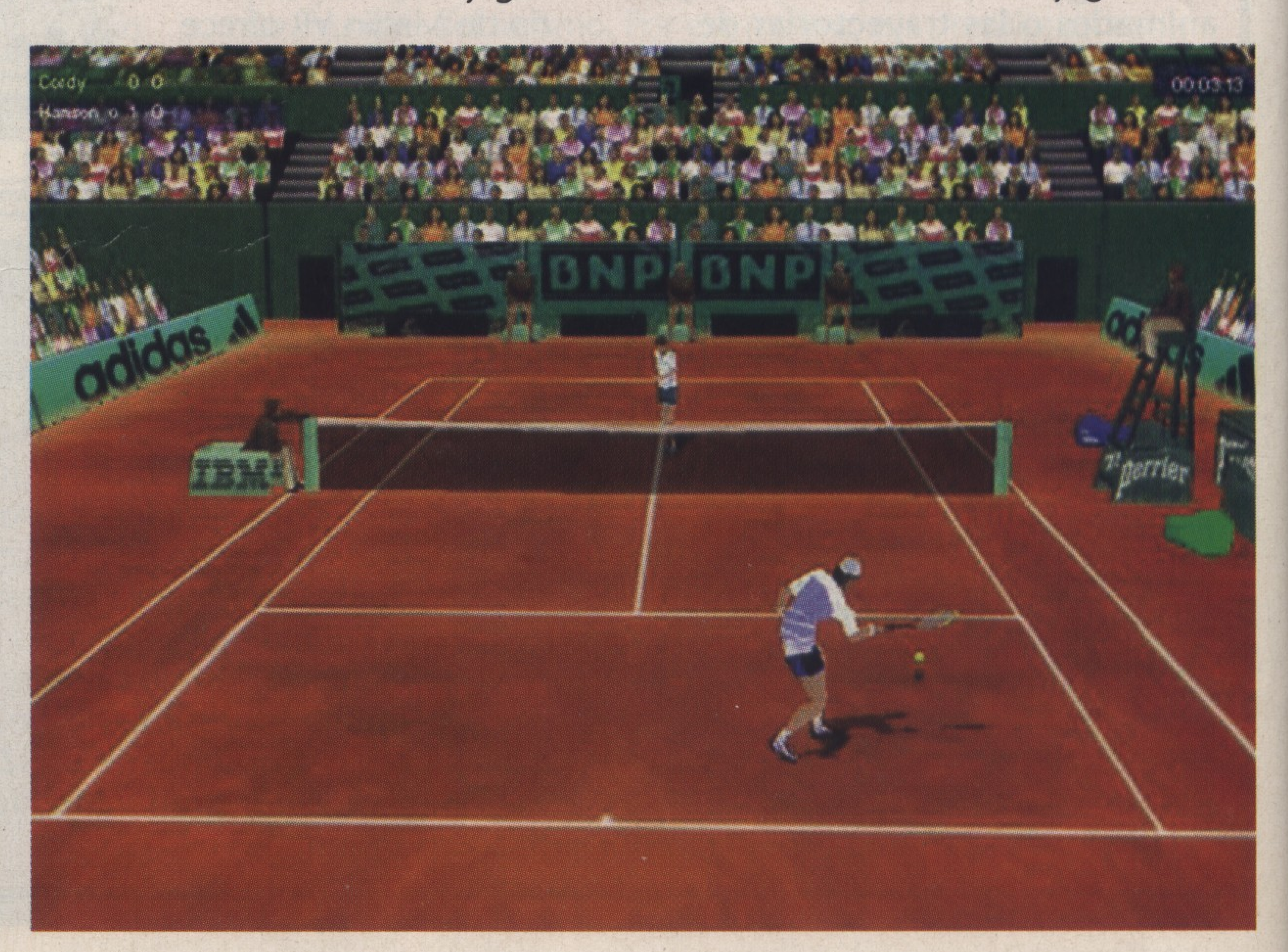
Con este juego podrás disputar partidos de tenis memorables contra el ordenador o contra otros jugadores. El juego aporta todo el realismo posible: drives, reveses, voleas, dejadas, globos, bolas liftadas, bolas cortadas, mates.. además de 50 jugadores diferentes, 16 pistas con todo tipo de superficies, 12 cámaras y ángulos de vista y un modo de repetición.

### Expendable

Se trata de un interesante juego de acción 3D. Se desarrolla en tercera persona en un contexto espacial donde se recrean excitantes mundos tridimensionales. Al mando de un comando expendable, compuesto por máquinas de matar que no tienen sentimientos, deberás desembarcar en el planeta Novocastria para averiguar por qué han desaparecido los colonos. Algunas de las características de este programa son su gran jugabilidad y nivel gráfico, la gran cantidad de potentes armas, el sonido espacial 3D y sus destacados efectos especiales de luz. Eso sí, como ya viene siendo habitual en este tipo de juegos, requiere una tarjeta aceleradora.

### Settlers III

La tercera versión de este juego ha incorporado muchas de las sugerencias recibidas de sus jugadores.








Así, ahora incorpora nuevos gráficos 3D con nuevas animaciones de todos los personajes, entornos, edificios y otros detalles. Asimismo se ha dado más importancia a los elementos estratégicos, teniendo ahora la estrategia tanta importancia como la construcción.

Settlers III contiene tres campañas diferentes: Romana, Egipcia y Asiática. Como en las anteriores entregas, el objetivo del juego es crear una civilización. En Settlers III, sin embargo, el jugador puede escoger las razas que desee usar. Éstas varían físicamente y llevan ropas diferentes, viven en casas distintas y construyen templos especiales para sus dioses. Dependiendo del dios al que esté adorando, la gente tiene diferentes características y habilidades. Hay más de 30 tipos diferentes de personajes que pueden seguir tus órdenes.

## Saga. La Furia de los Vikingos

Durante tres siglos el mundo conocido miraba hacia el horizonte esperando no verlos aparecer... Durante tres siglos su nombre fue sinónimo de poder y coraje, pero también de brutalidad y destrucción... Durante tres siglos surcaron los mares con una sed de conquistas imposible de saciar, dejando tras de sí un rastro de muerte y destrucción. Saga es el juego de estrategia en tiempo real que te situará justo en medio del universo de los Vikingos. Crea sus clanes, gestiona sus recursos, crea tus armas, construye tus naves Drakkar y prepárate para la conquista de otros pueblos.

Saga dispone de hasta siete razas diferentes (vikingos, gigantes, trolls, elfos, centauros...) cada una con sus propias características, un gran realismo total (condiciones atmosféricas cambiantes, cambio de estaciones, cambio en la dirección del viento...), diferentes categorías de magia, posibilidad de juego en red y una banda sonora con música celta.

## Deer Hunter II

Friendware presenta la siguiente entrega de uno de los juegos con más éxito de ventas en Estados Unidos de todos los tiempos. Se trata de un simulador de la caza del ciervo. La principal evolución respecto al primer Deer Hunter está en que ahora el cazador se encuentra en un entorno realista tridimensional por el cual se puede mover a la búsqueda de su pieza. Incluye todo lo que un cazador profesional requeriría para una




simulación realista: nueve tipos de armas, sala de trofeos, utensilios de cazador (cebos de olor, camuflajes de olor corporal, imitación de la llamada del ciervo, unidad GPS, ruido de cuernas, prismáticos, decorados de ciervos, etc.) y doce escenarios donde cazar esa codiciada pieza.

## African Safari Trophy Hunter

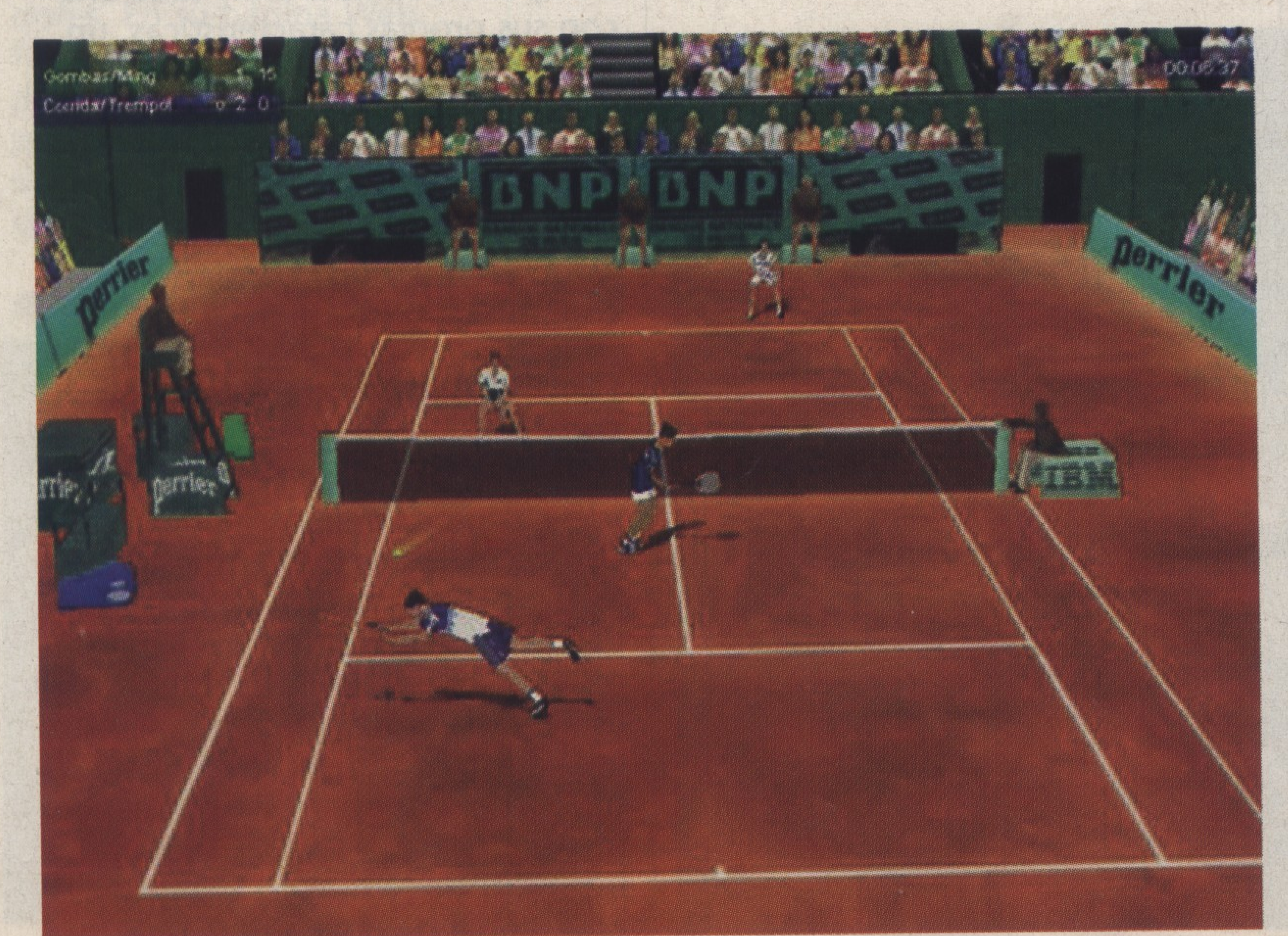
Otro simulador de caza que te lleva a un entorno tridimensional donde disfrutarás de un peligroso viaje a través de las impresionantes sabanas de Africa, en búsqueda del trofeo más preciado. Siente tu pulso acelerarse mientras sigues los rastros de tu pieza en su majestuoso entorno; nota como fluye la adrenalina cuando de repente se giran hacia ti, preparados para atacarte. Éste es un juego de caza 3D de increíble realismo, peligro e intriga. Algunas características del juego: gran variedad de animales salvajes (elefantes, cebras, kudus, elands, leones, buitres, cocodrilos), magníficos escenarios (reservas de caza de Zimbabwe), muchas armas (rifle de cerrojo con mira telescópica, rifle de gran calibre con doble cañón) y accesorios para todos los gustos (prismáticos, brújula, indicador del cambio de dirección del viento, cámara de fotos).

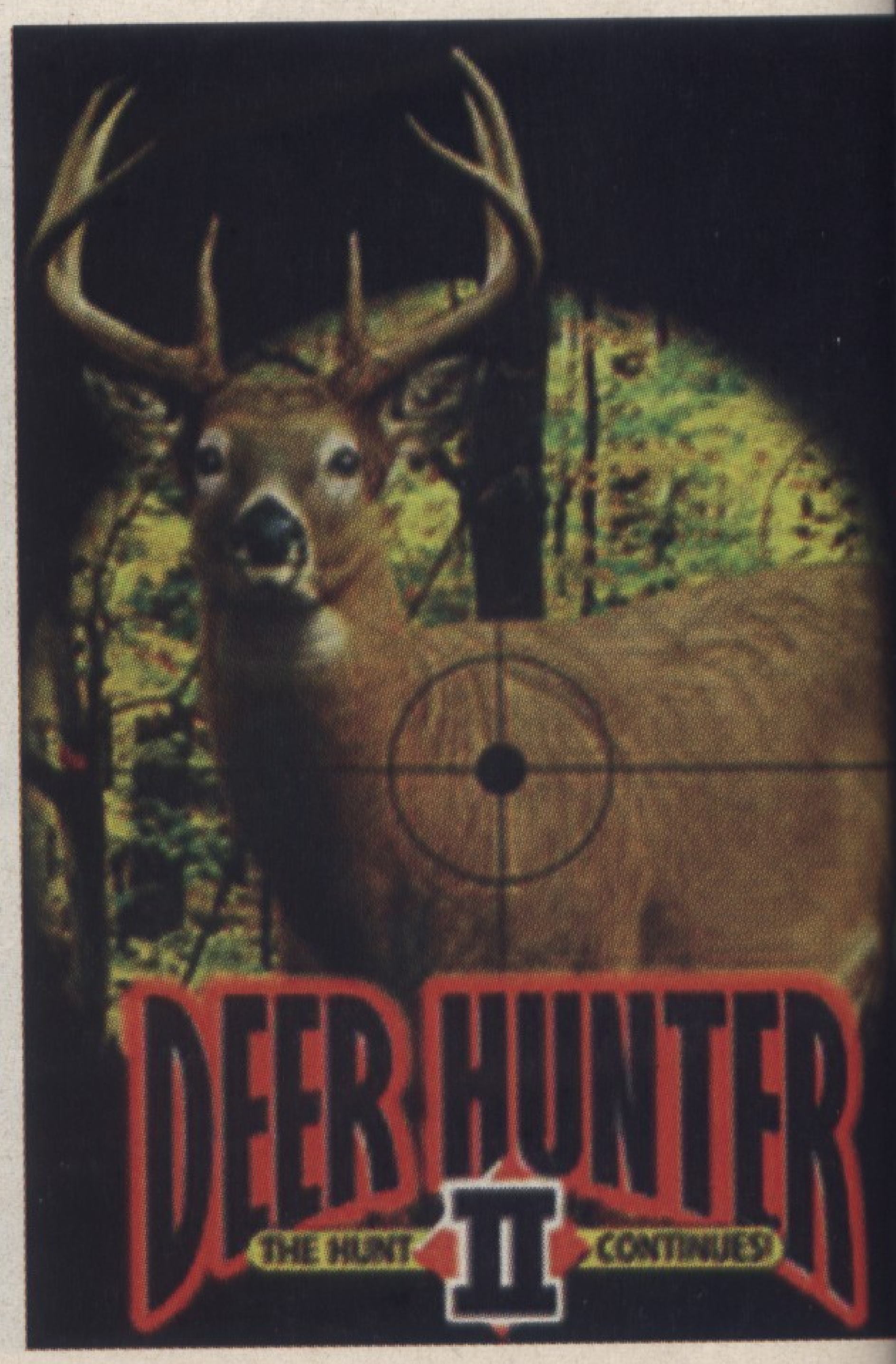
## Próximamente

Friendware tiene guardados en su cartera algunos títulos que se presentan más que interesantes. Entre ellos Blade, un juego largamente esperado, Guardian y Devil inside, dos interesantes programas a caballo entre el rol, la aventura y la acción.







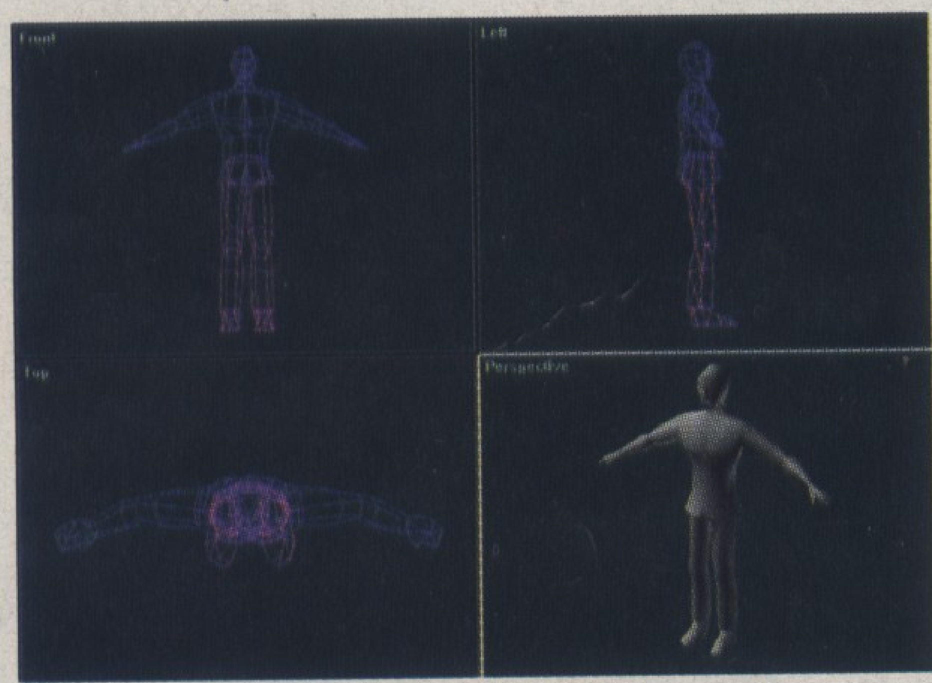

¡OKENKA!
O K E N K A I
JENNIFER
Azafata infiltrada
ROJAS
Narcotraficante
VINCENT
El brazo ejecutor
Dr. STRAUSS
Drogas sintéticas
BUCKY
Maleante portuario
SMITH
Camello internacional
Estrategia explosiva en
un futuro demoledor
JOSHUA X
Mercenario
ANTIVICIO
¡PREPARATE!
MISSION ONE
PC CD rom
COMPATIBLE
WINDOWS 98
W95
HAMMER Technologies
HAMMER Technologies
C/ Alfonso Gómez, 42, nave 1-1-2
28037 Madrid (Spain)
Tlf. (91) 304 06 22
Fax. (91) 304 17 97
e-mail: hammert@studios.com

# Ayer y hoy en la programación de Videojuegos

## Las herramientas del programador

**Todos aquellos que llevamos cierto tiempo en el mundo del PC hemos visto que la evolución de los juegos ha sido pareja a la del propio PC. Ciertamente, hasta hace bien poco los juegos iban por detrás de las máquinas; pero actualmente esto ya no es así, de hecho los juegos superan los requerimientos de las máquinas con demasiada facilidad.**

Lo que anteriormente se programaba en ensamblador, obteniendo el tope de las máquinas, ahora se realiza en lenguajes que no alcanzan en velocidad o potencia al lenguaje ensamblador. A cambio de esta merma, el programador gana en velocidad de desarrollo y en facilidad de programación, porque, seamos sinceros, a pesar de su extraordinaria potencia, programar un juego actual en ensamblador puro y duro es una tarea imposible.



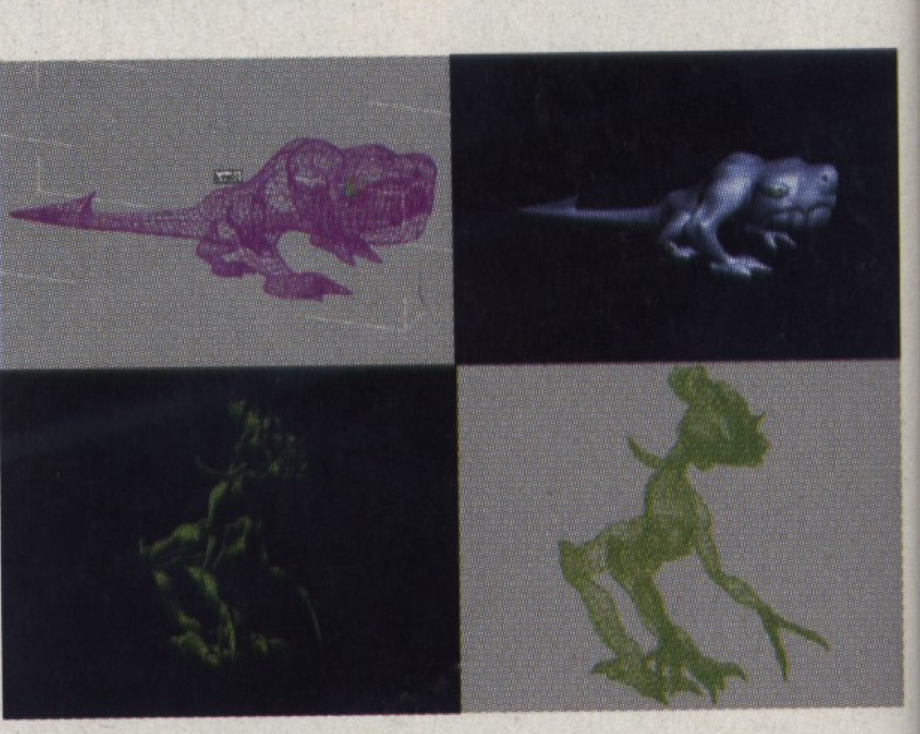
**Modelo en baja de 650 facetas.**

Sólo es permisible utilizar ensamblador para rutinas concretas, integradas en el conjunto del programa en C++. De hecho, la mayor parte de los juegos actuales se desarrollan en lenguajes de alto nivel, principalmente C++. Éste nos proporciona una gran potencia además de una programación cómoda y ágil.

Los sistemas también han evolucionado. Hemos pasado de pelearnos con el MS-DOS a desesperarnos con Windows. Para el programador, las cosas casi se han complicado un poco más en vez de mejorar. Todo queda más bonito, más elegante, pero hay que reciclarse y aprender cosas nuevas. Programar juegos para Windows ha pasado de ser imposible a ser factible con las DirectX. No son del gusto de ningún programador (OpenGL es infinitamente mejor) pero no nos queda más remedio que pasar por el aro de Microsoft si queremos ver nuestro juego en todos los ordenadores del mundo.

¿Adónde queremos llegar con esta introducción? Pues al hecho de que la programación de videojuegos es una tarea ardua, compleja y no exenta de desilusiones. El que desee programar un juego de forma profesional debe estar dispuesto a saber de todo, a programar en casi todos los lenguajes y a estar al día en todo lo relacionado con la programación gráfica.

El objetivo de este articulo es repasar las diferentes herramientas utilizadas en la programación de videojuegos, desde los principios del PC y hasta el día a día que nos toca vivir a los sufridos creadores de juegos. Podríamos dar un repaso a las herramientas usadas en los tiempos del Amstrad y del Spectrum, pero os puedo asegurar que no llenaríamos ni media línea, aquellos tiempos eran los del ensamblador puro y duro.



**Modelos en alta.**

### Primeros pasos

Como todo comienzo, la programación de juegos para los primeros PC era una pesadilla. Conocer los entresijos del MS-DOS y de la maquina eran cuestiones obligadas. El MS-DOS, entorno totalmente carente de ventanas, nos obligaba a programar todo el sistema multimedia necesario para usar nuestro juego. Los juegos más potentes se programaban ensamblador, naturalmente. Una labor realmente compleja pero que lograba un resultado inigualable.

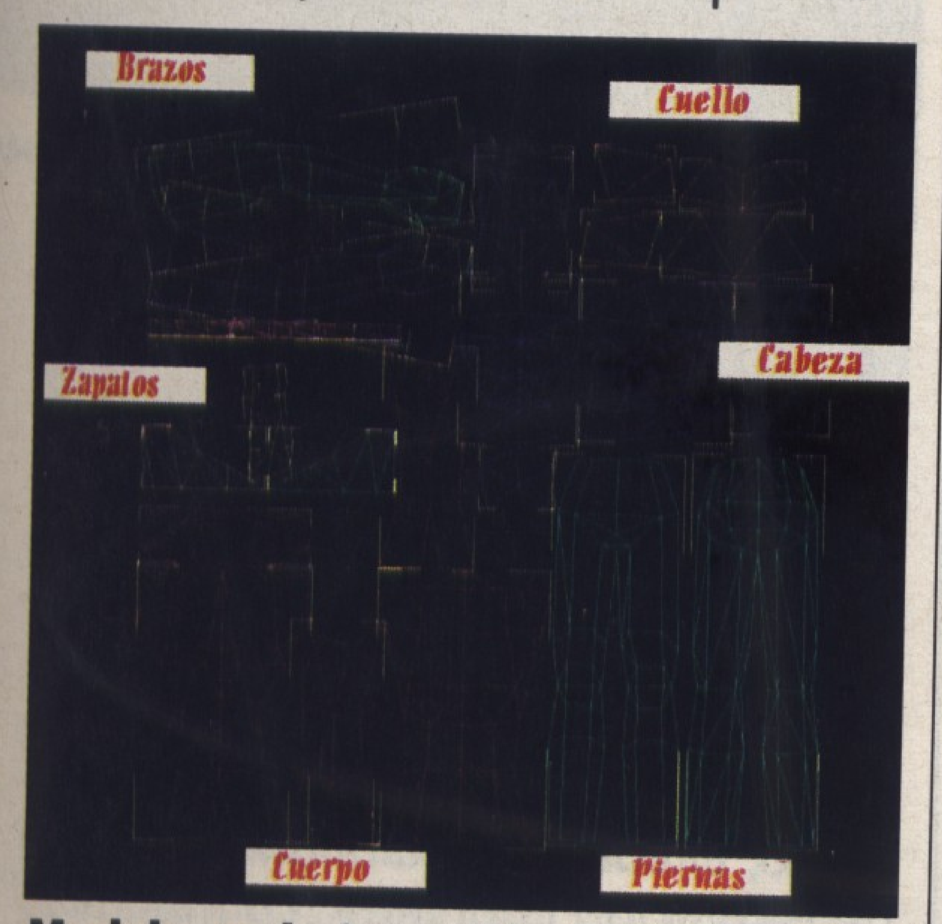
Normalmente se utilizaban el Turbo Assembler y el Macro Assembler. Ambas herramientas incorporaban entornos integrados en sus ultimas versiones que hacían que pudiéramos prescindir de los editores separados del entorno. Sin embargo, en muchos casos era más fácil utilizar nuestro editor habitual y compilar desde la línea de comandos. Lo cierto es que no eran productos muy agraciados, sobre todo el Macro Assembler.

### C++ al poder

Poco a poco, según los proyectos se fueron haciendo desmesurados, los programadores fuimos migrando al potente C. Aunque los primeros compiladores no permitían mezclar el C con el ensamblador, poco a

poco incorporaron esta propiedad. El Turbo C, de Borland, fue uno de los compiladores más utilizados. Un entorno bastante agradable, una ayuda breve pero eficaz, un gran producto en general. El C dejaba al programador realizar una programación más compleja, permitiendo que los juegos crecieran en posibilidades.

Los productos de Borland siempre han gozado de una gran reputación entre los programadores. Las empresas sin embargo se han decantado por los productos Microsoft. No es que los productos de Microsoft sean mejores, en muchos casos son netamente inferiores, pero las diferentes dificultades por las que ha pasado Borland, hacían que muchas empresas se decantaran hacia un producto con futuro y soporte técnico asegurados. Prácticamente todos los programadores de la vieja escuela hemos pasado

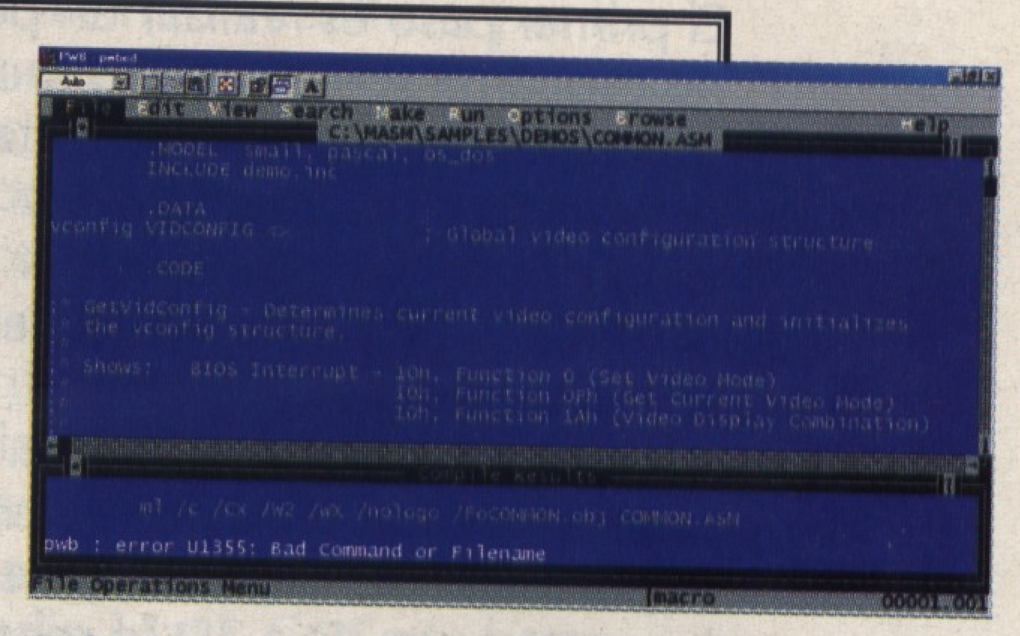



**Modelo en baja con las facetas separadas para la elaboración de la textura.**

por el Turbo C++, un gran producto, sencillo pero altamente eficaz. El único problema que tenía era obtener la memoria necesaria para los programas. El concepto de memoria virtual era una teoría para el MSDOS; obtener algo más de los 640K de memoria base era una tarea titánica. Hasta la llegada del Watcom.

## Macro Assembler

Cuando Borland era rey indiscutible entre los programadores, hizo su aparición un nuevo ensamblador dispuesto a reivindicar el nombre de Microsoft. Sin llegar a alcanzar el podio, ocupó un lugar entre los compiladores por sus herramientas para la depuración. Su entorno no era agradable para programar pero la depuración, factor dominante de la programación en ensamblador, era un aspecto que este producto exprimía con rotundidad.



**Mientras que su entorno era demasiado estático e incómodo, Macro Assembler destacaba por tener una gran capacidad para la depuración.**

## Watcom y la memoria

Watcom supuso una revolución. Utilizando una librería de ejecución, este compilador nos proporciona memoria virtual, es decir, toda la memoria que con nuestro disco duro podamos necesitar. Podemos decir que hay un antes y un después en la programación de videojuegos, y Watcom es la frontera. Sin embargo el declinar de Watcom llega con la aparición de Windows 95.

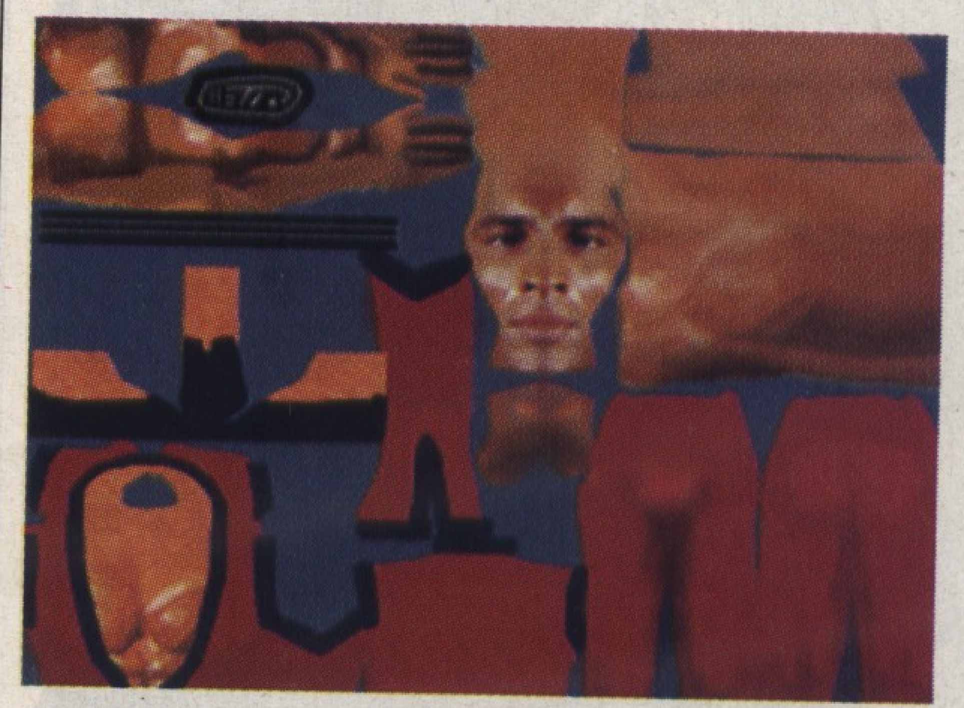
Con Windows 3.1 ya teníamos memoria virtual, pero programar juegos bajo Windows era algo imposible. Sin embargo, con Windows 95 y la aparición de las DirectX, la programación en Windows de rutinas gráficas potentes fue un hecho.

No es que no se pudieran utilizar las DirectX con Watcom, se puede, igual que se pueden utilizar con Delphi, pero seamos serios, hoy por hoy, para programar con eficacia un producto que vaya a utilizar rutinas de Microsoft, la herramienta es Visual C++. No es que sea lo mejor del mercado pero se integra a la perfección con las DirectX, así como con las API de Windows; no en vano son productos de la misma casa.

## Visual C++

Está lleno de defectos, así como de virtudes. El entorno es bastante cómodo de usar aunque es mejorable, como todos los productos Microsoft. La ayuda es bastante completa pero está plagada de errores. Las MFC son una burda copia de las versiones de Borland de lo que debe ser la programación visual. La ayuda contextual es otra copia de Borland. Quizás sea que los señores de Borland tienen las ideas y los de Microsoft los medios.

Sobre diez, personalmente le asigno un siete, lo cual es ciertamente un halago para un producto de Microsoft. Hasta la fecha, pocos productos de esta casa dan la talla; mucho tienen que aprender los señores de Microsoft de compañías como Borland o IBM.



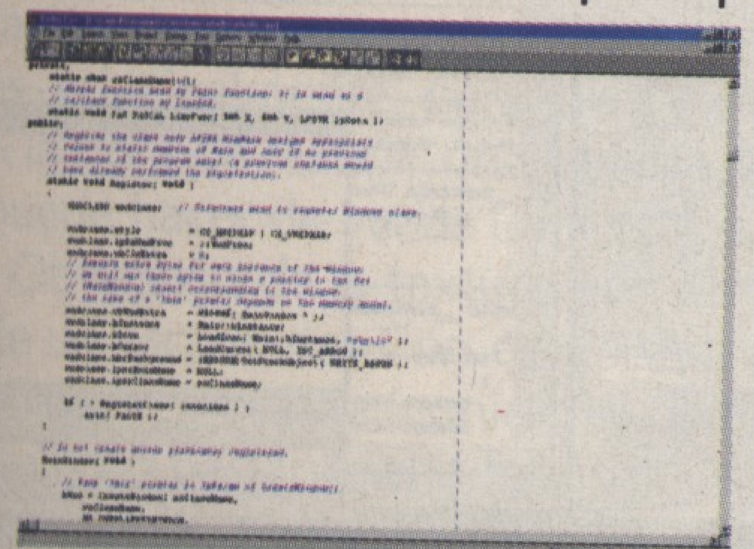
**Textura creada con programa de retoque fotográfico.**

Actualmente, la versión 6.0 del compilador Visual C++ es la utilizada en el mundo de la programación de videojuegos. Todos hablamos mal de Microsoft pero no dejamos de utilizar sus productos. Es un hecho, desde el usuario hasta el programador, todos usamos Microsoft.

De la misma forma que los juegos han ido enriqueciéndose gracias a las facilidades proporcionadas por los distintos compiladores y sistemas operativos, no seria justo negar la importancia de la evolución de los lenguajes. No deja de utilizarse el lenguaje ensamblador, pero éste ha dado paso a la facilidad de uso y versatilidad de lenguajes de alto nivel como el C. Hoy es su evolución, el C++ (C orientado a objetos), el que ha tomado el relevo, el rey actual de la programación.

## Turbo C++

Este compilador de Borland fue uno de los primeros en abordar la programación en C++. De hecho, puede que fuera el primero que entendiera realmente la revolución que supuso la programación orientada a objetos.



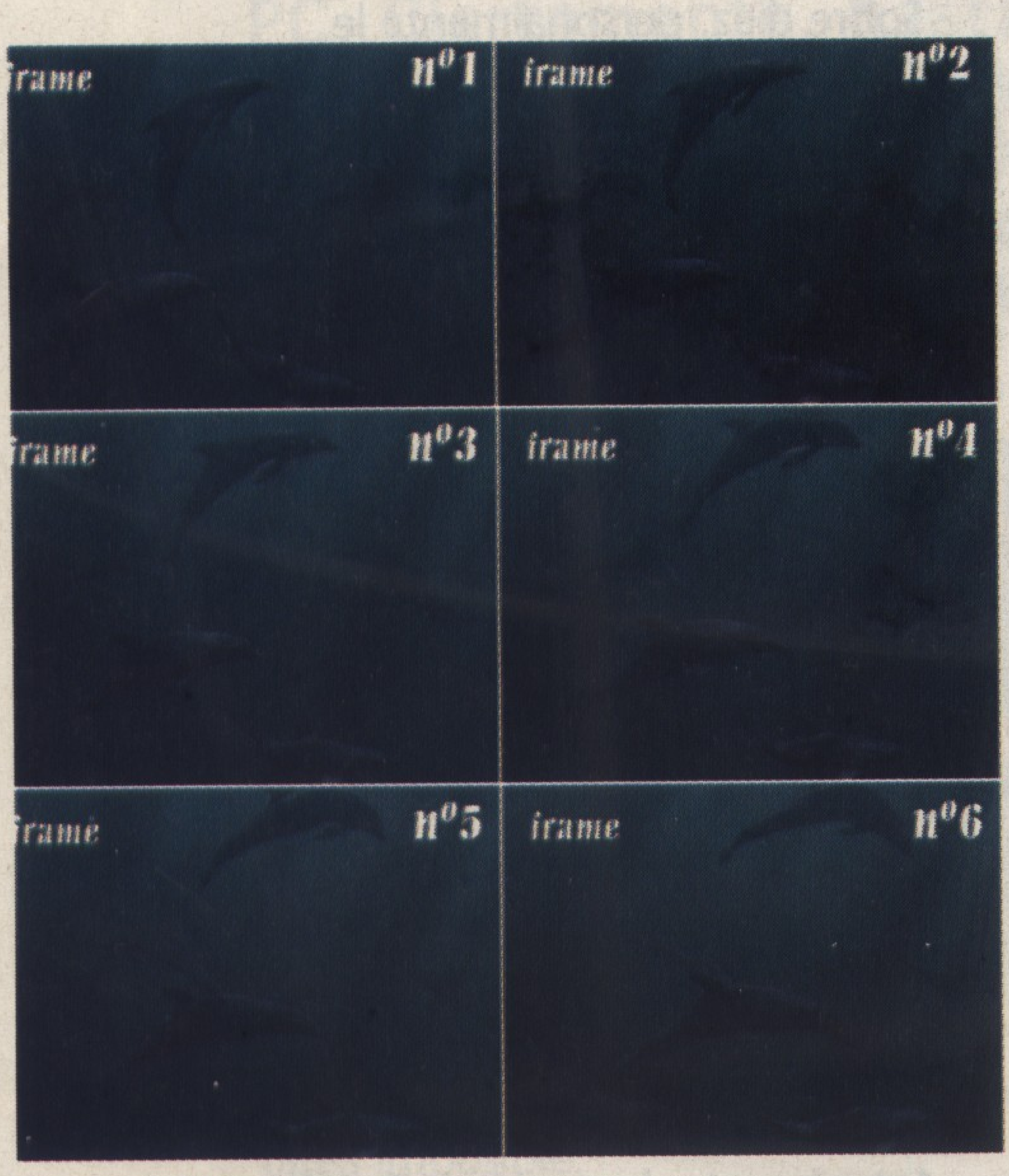
**Las versiones para Windows de este magnífico compilador no hacen justicia al magnífico producto que Borland diseñó para MS-DOS.**

Las versiones iniciales, obviamente para MS-DOS, tenían un entorno bastante configurable y una ayuda excelente. La aparición de Windows, trajo consigo la evolución de una gran serie de compiladores de C++, preparando Borland la versión de este clásico para Windows. Este producto no gozó de gran aceptación en su momento. El predominio del MS-DOS como plataforma de arranque de los juegos hacía de este compilador orientado a Windows, un maldito en este campo.

## Creación de modelos 3D

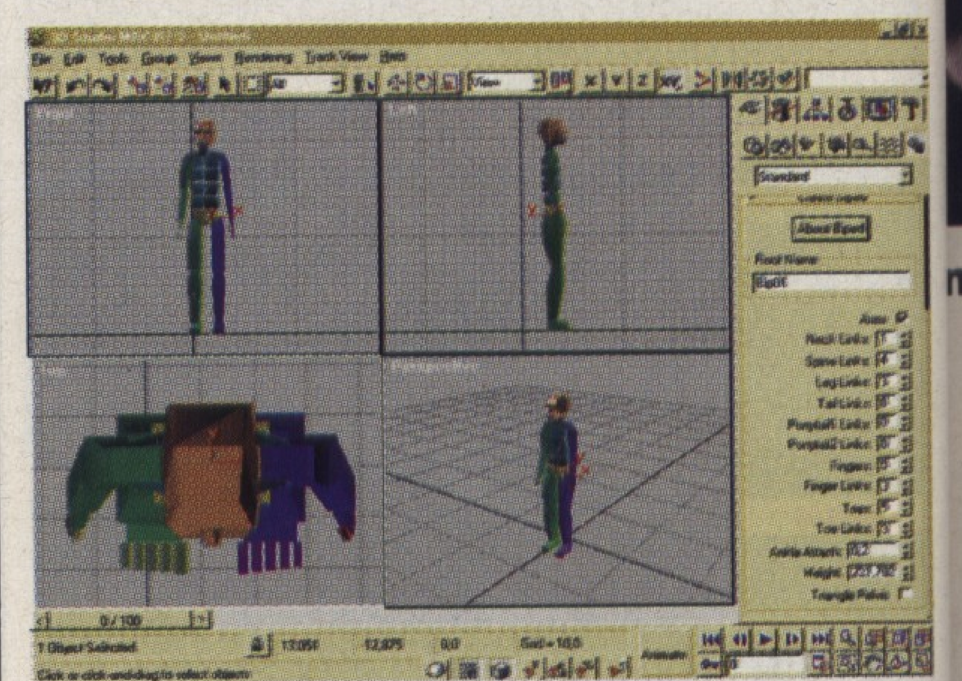
El primer paso es realizar un pequeño esquema sobre papel o mentalmente, además de documentarnos en todo lo necesario. Una vez hecho esto, nos ponemos a modelar el objeto o personaje teniendo en cuenta una serie de pautas:

- **Estudio del nivel de detalle.** Este factor depende principalmente de la importancia del elemento y de si el juego va a ser 2D o 3D (o mezcla de ambos). Si el juego es 3D en su totalidad, tendremos que escatimar en polígonos para tener un juego fluido; sin embargo, modelando para un juego en 2D el problema de los polígonos no existe ya que la malla modelada no se exporta a la programación, únicamente sus renders de movimiento y acción. Entre estos dos sistemas las diferencias son notables. Un juego donde intervengan una cantidad elevada de personajes no se puede realizar por el momento con modelos 3D, por la simple razón del gran numero de polígonos con que necesitaríamos contar en la escena.



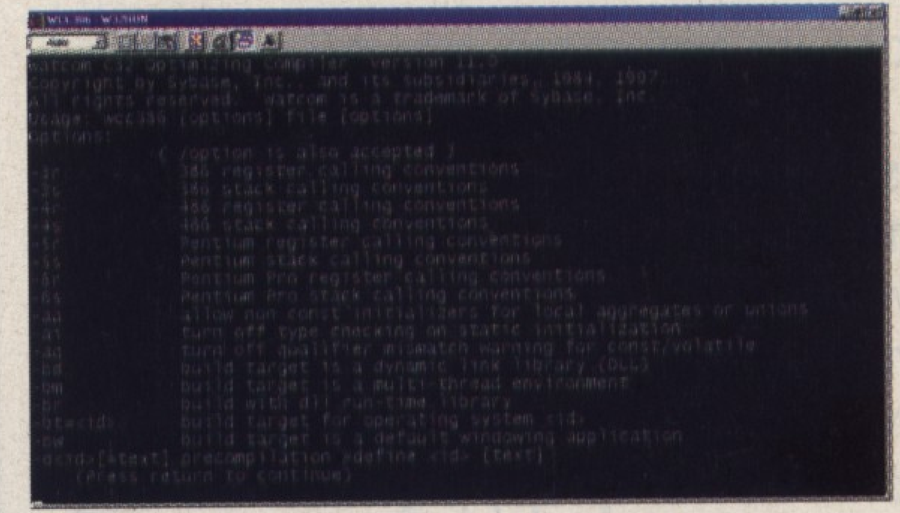
**Animación frame a frame.**

- **Adecuación de los elementos referentes a los tamaños.** Es un elemento a considerar. En todos los objetos pequeños se tiende a exagerar el volumen para que, de esta forma, se pueda apreciar sin problemas y se distingan de otros elementos colindantes; justamente lo contrario les ocurre a los elementos de la escena de gran tamaño.
- **La iluminación.** En un juego de entorno 3D es proporcionada casi siempre por programación, con lo cual es un factor a descartar, aunque no es un elemento complejo. En cambio, si nos desenvolvemos en un entorno 2D el tema de la iluminación cobra importancia ya que tenemos que simular todos los brillos y sombras necesarias, para cuando haya que preparar la animación que se realizará utilizando los frames de la animación.
- A la hora de aplicar las texturas hay que diferenciar una vez más a qué tipo de juego va destinado. Si va ser un juego en 3D la textura aplicada debe de acompañar en otro fichero tipo gráfico como bmp, pcx.., dependiendo del motor 3D del juego, el cual soportará unos tipos de resolución concretos. Lo más normal hoy en día es 256 y 512, además de otros tamaños que dependen de su cometido. Por tanto, tendremos dos ficheros: la malla y la textura. En cambio, si va destinado a un juego 2D no se requiere el uso de estos dos ficheros ya que la animación consiste en imágenes una tras otra. Otra cosa importante es la adaptación de las texturas, deben estar acondiciona-


**Biped (Character studio).**

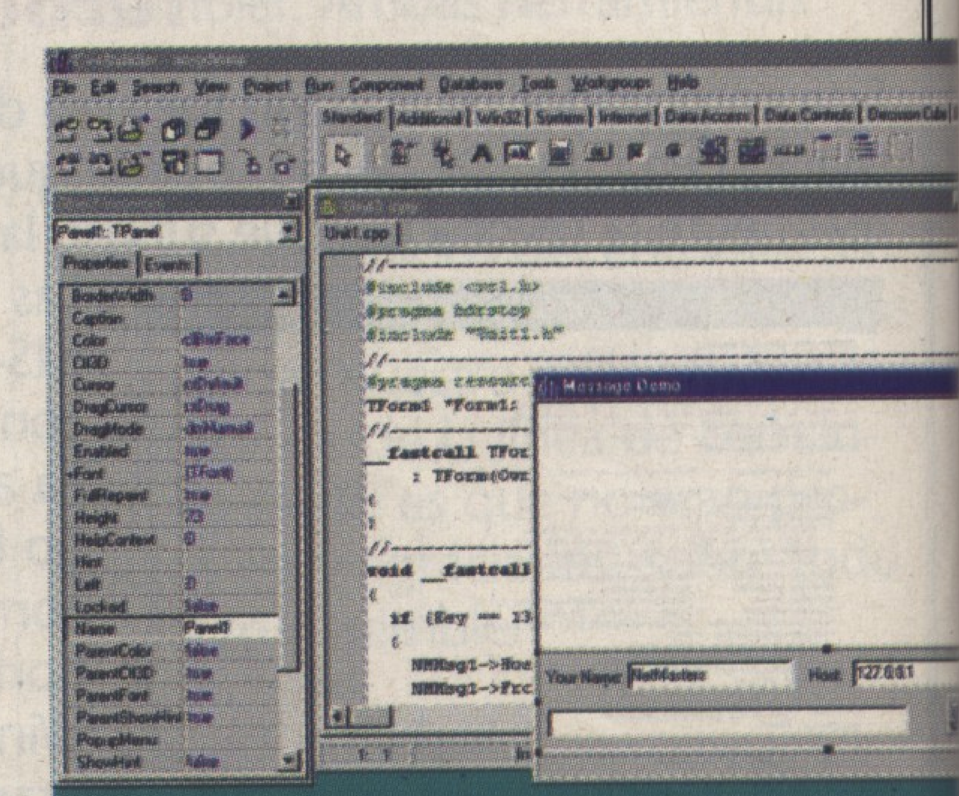
### Watcom

El MS-DOS no era un lecho de rosas para los programadores y mucho menos para los creadores de juegos. Los videojuegos siempre han sido grandes devoradores de recursos, principalmente de memoria, y el MS-DOS no estaba dispuesto a entregarla con facilidad, de hecho, arrebatar memoria al sistema operativo era una labor tanto del usuario como del programador, debiendo en muchos casos llegar a un consenso para decidir cómo configurar el sistema para lograr que el juego por fin arrancase.
Watcom abrió la ventana de la memoria virtual: una simple librería linkada al proyecto y podíamos tener toda la memoria que nuestro equipo en combinación con el disco duro pudiera ofrecernos. Para los programadores, fue un rayo de luz entre las tinieblas. Watcom era el compilador de videojuegos por derecho propio.


**A pesar de ser un compilador en línea, sin entorno integrado, supuso una revolución en el mudo de los videojuegos.**

### Borland C++ y Borland C++ Builder

Aparentemente, éste es un compilador más. De hecho no es uno de los mejores de esta casa, puesto que los compiladores de C++ para Windows de Borland sólo han empezado a alcanzar un merecido prestigio entre los programadores cuando han llevado a la máxima expresión la programación visual, y esto sólo lo han alcanzado ahora.
Borland ha tomado el concepto de programación visual de su compilador de Pascal para Windows, Delphi, y lo ha llevado al mundo de C++, con total trasparencia para el programador y sin recurrir a complicadas soluciones que confundan al programador. Las versiones del Builder son realmente potentes y sencillas de usar; todo lo contrario que sus competidores más cercanos en lo referente a programación visual en C++. Quien haya programado en Delphi conoce el placer de programar visualmente. Nos olvidamos de realizar complicadas llamadas a las API y preparamos el codigo necesario para responder a los eventos de Windows. La programación visual llevada a la máxima potencia. Nos es la mejor herramienta para programar bajo DirectX, pero es mucho más que Visual C++ y sus temidas MFC.


**La programación visual se llama Bor[...] Tanto en Delphi como en Builder C+[...] lizamos una programación totalmen[...] visual y sin obstáculos.**

## Visual C++

Y llego el producto estrella de Microsoft, Visual C++, un gran compilador, excelente entorno integrado, gran ayuda, buenas herramientas, facilidad de uso, pero nada de visual. ¿Por qué llamar Visual C++ a un producto que no ofrece ayuda al programador? Sí, es bonito y potente, pero ¿nos ayuda a diseñar visualmente nuestro programa? Las MFC más que un acierto parecen un desatino, no son nada sencillas de utilizar, sin embargo ahí están, cada cual es partidario de usarlas o no pero, hoy por hoy, los programadores de videojuegos suelen usar con rotundidad las API de Windows para diseñar el entorno de una aplicación.

**Visual C++ 6.0 nos ofrece un entorno fácil y cómodo de usar, algo imprescindible para programar productos complejos.**



**Sensores de captación de movimiento.**

das para que no se produzca tileo por la repetición de la imagen. Todas estas texturas se realizarán mediante un programa de retoque fotográfico o de diseño gráfico.

- **Preparación de la geometría o malla para el proceso de texturización.** Esta preparación de la malla se refiere a los juegos 3D. Hay diversas formas pero la más extendida, y también más laboriosa, es, una vez concluido el modelo, separar las caras de la geometría y colocarlas en grupos situados de forma lo más ortogonal posible a una de las vistas. Con ello sacaremos un render de una de las ventanas y, con ese render o captura, podremos crear nuestra textura, que luego aplicaremos con gran exactitud ya que tenemos la geometría representada.
- También existen varios plug-ins para texturizar y retocar la textura una vez aplicada en la malla; por ejemplo 4D Paint.

## Elaboración de las diferentes texturas

Ya tenemos el render con la delimitación de nuestra malla (este paso es, principalmente, para juegos 3D). A la hora de realizar una textura, se puede hacer de varias formas:

- Obtención de la textura partiendo de un escaseado. Con este sistema obtendremos una iconicidad con un realismo bastante alto, consiguiendo siempre lo que pretendamos. Otra forma es generar la imagen a base de pintar, crear brillos, sombras y demás decoraciones.
- Cuando tengamos terminada la textura a nuestro gusto y acorde con el tipo de juego, nos la llevaremos a nuestro programa 3D en el que fabricaremos un material para poder aplicar nuestra textura.

Esto que os hemos comentado de forma breve y esquemática es la manera de entrelazar los dos tipos de programas; tanta importancia tiene uno como el otro.

## Métodos de animación

Una vez terminado nuestro modelo, perfectamente texturizado, y según sea nuestro tipo de juego, necesitaremos realizar unos pasos que se irán repitiendo con todos los elementos animados, los que tenemos que preparar para la realización de todos los gráficos.

- **Frames.** Este método se realiza para la elaboración de sprites. Es decir, el movimiento se realiza mediante un número de imágenes consecutivas a




**Cámara especial de captación de movimiento.**

cierta velocidad. Esto normalmente se utiliza para juegos 2D, aunque los juegos 3D también lo hacen con bastante frecuencia. Con este método obtenemos una animación del genero 2D. Estas imágenes pueden ser elaboradas mediante un programa 3D, en el cual tenemos que animar nuestro objeto y luego crear un numero de frames correspondiente a la animación creada, o bien con un programa 2D de diseño gráfico o retoque fotográfico.

- **Animación 3D.** Este sistema es uno de los más atractivos, por lo menos desde mi punto de vista, ya que los pasos realizados hasta llegar aquí son bastantes numerosos. Lo más complejo es animar al ser humano y cuerpos orgánicos; sucede lo contrario con elementos mecánicos, los cuales giran, se mueven y articulan siempre por el mismo sitio.
- **Utilización del Character Studio (3DStudio Max).** Con esta herramienta, se simplifica bastante el trabajo de animación de nuestros modelos. El problema de este tipo de herramienta es que a simple vista no tienes que trabajar mucho con la animación, pero el toque humano para el proceso de animación se caracteriza por ser una cosa personal. Podemos realizar una infinidad de animaciones con el ABiped@ con el cual tenemos un sistema de jerarquías ya establecido. Lo adaptaremos a la malla escalando sus diferente partes. Una vez terminada la operación le aplicamos el modificador APhysique@ con el cual nos aplicará las deformaciones, en articulaciones y en los diferente pliegues de nuestra malla.

**Equipo de motion capture.**

- **Motion capture.** Este sistema de animación es el más realista ya que lo realiza mediante un sistema de captación de movimientos usando unos sensores, los cuales van situados en la persona u objeto que realice el movimiento; estos se ven reflejados en un sistema de coordenadas tridimensional. Estos movimientos son captados mediante cámaras; dependiendo de la empresa se suelen utilizar alrededor de ocho cámaras para la captura de movimiento. Después de utilizar este gran despliegue de equipo, es necesario pasarlo por un programa específico para poder utilizarlo con los programas de animación 3D más conocidos (3D Max, Softimage, Lightwave, Maya, Alias...). Este proceso convierte el movimiento en AKeys A (llaves de animación, puntos clave de los movimientos). Estas animaciones se integran en los programas citados. En ellos podemos seguir retocándolas y darlas el toque definitivo.

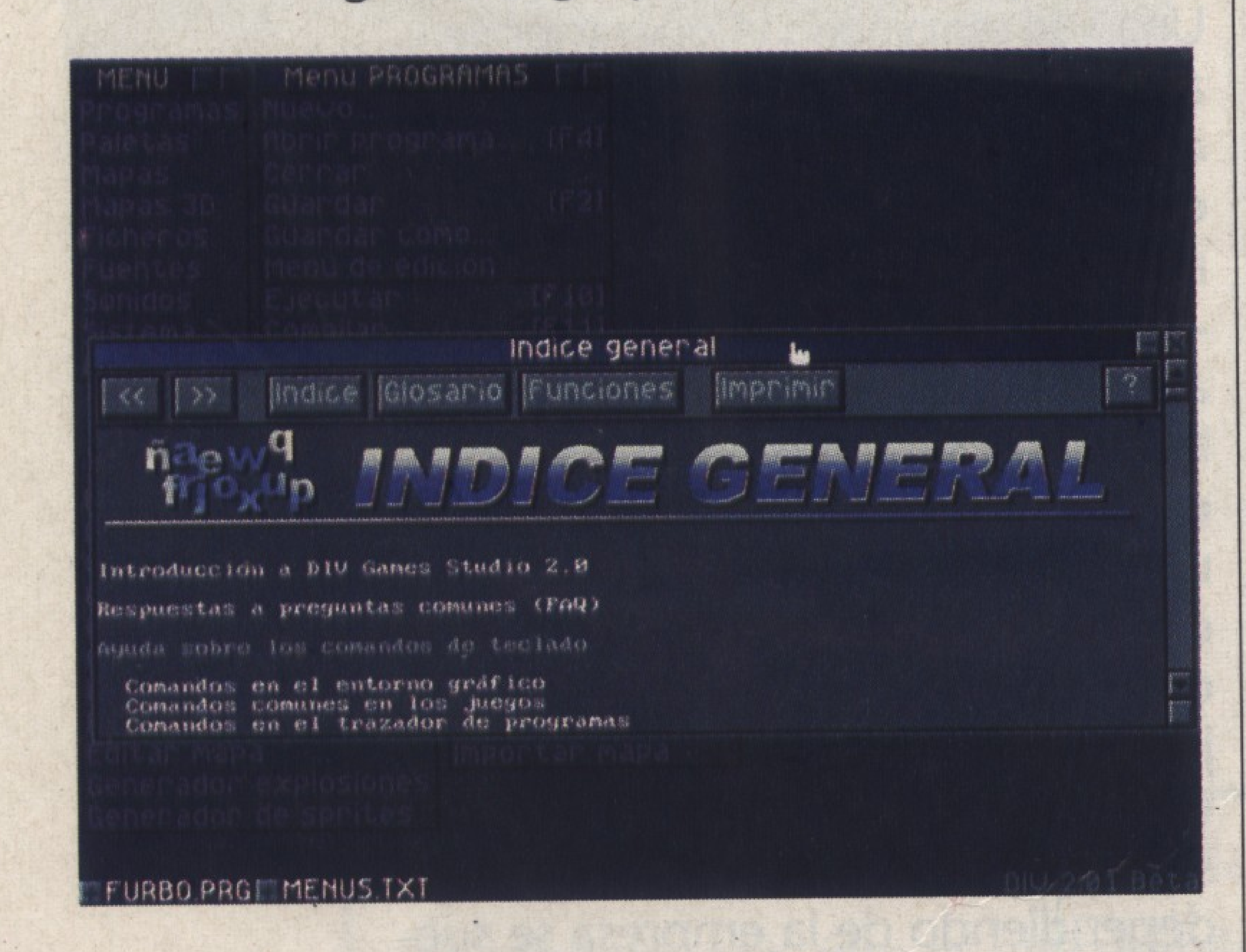
**Armando Vélez-Raúl Bravo**

# Comenzando a trabajar

## Las primeras dudas

**DIV2 ya esta a la venta desde hace algún tiempo. Mucha gente ya ha empezado a tratar con la nueva herramienta y, como suele suceder en este tipo de experiencias, las preguntas y problemas aparecen en gran cantidad. No hay problema, desde estas líneas veremos los nuevos aspectos y las dudas que pueden plantearse.**

Hay bastantes novedades en esta nueva herramienta, y aunque se ha intentado simplificar al máximo el uso de todas ellas, no siempre se consigue. Por eso vamos a dedicar este articulo a las primeras dudas que se han creado al iniciar el trabajo con DIV2. Las repartiremos en pequeños apartados, asociados por la parte novedosa donde se sitúan. Además, repartiremos todo en dos grandes grupos: el entorno y el




lenguaje. Dentro del entorno veremos todas las herramientas nuevas de las que dispone, incluyendo también las mejoras en el editor gráfico. Por el lado del lenguaje, veremos las nuevas funciones, que no son pocas. Pero sin más preámbulos, metámonos en faena.

### El entorno

Es lo primero que vamos a ver. Únicamente haremos hincapié en las partes novedosas, obviando todo aquello que estaba en la antigua versión. Se han separado por herramientas, es decir, el generador de sprites, el generador de mapas 3D, etc. Pero empecemos con las dudas sin más preámbulos.

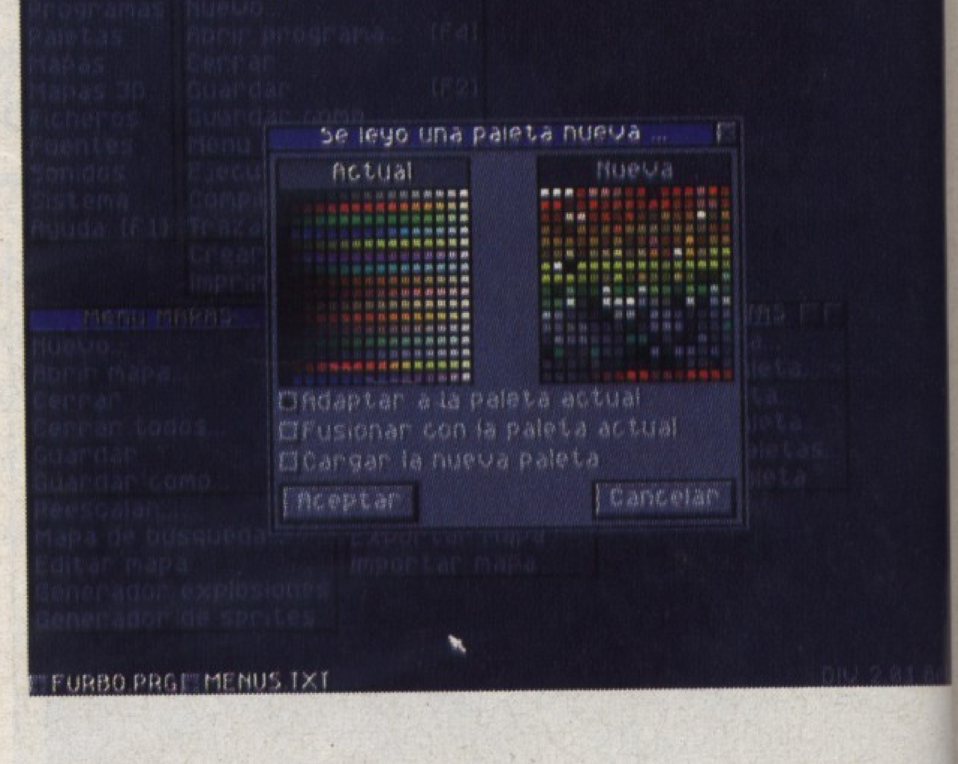
### Generador de sprites

Antes de empezar a hacer gráficos con esta herramienta, se debe organizar todo el trabajo. Es decir, se deben tener planeados antes la cantidad de gráficos que se van a necesitar, la posición de estos, el tipo de vista, etc. Esto, además de facilitar la generación de sprites, sirve para calcular aproximadamente la cantidad de memoria que vamos a necesitar, actuando en consecuencia.

Una de las cosas más importantes es la textura del personaje. Para cambiarla se debe hacer click con el ratón encima del mapa gráfico que la designa. Para ver cómo funciona, un ejemplo fácil y divertido es el de cambiar la cara de cualquiera de las texturas de ejemplo por una foto nuestra, y así podremos fardar con los amigos. Otro truco es el de cambiar la textura por una de color liso, y tendremos una especie de *Terminator,* aunque no sepamos su utilidad.

Una vez que tenemos la textura elegida, debemos elegir los movimientos. Existen gran cantidad de ellos en una lista. También podremos elegir la vista, bien introduciendo los datos numéricamente o usando el ratón. Si se pulsa con el botón izquierdo dentro de la ventana del personaje, podremos elegir la vista moviendo el ratón. Si por el contrario es el botón derecho el que se mantiene pulsado, podremos ejecutar la animación. También existe un botón para ir paso a paso en la animación.

Si queremos hacer un movimiento más rápido, deberemos reducir el numero de sprites. A mayor numero de sprites las animaciones son más bruscas, pero también más lentas. También podemos variar el tamaño de la animación, cambiando el valor del tanto por ciento.




Con esto se tendrían introducidos todos los valores necesarios. Deberíamos entonces generar ese movimiento y todos los demás movimientos y vistas que necesitemos. Hay que decir que siempre se puede manejar estos últimos a mano. Es decir, cuando se tengan todos los movimientos dentro de un FPG, se puede generar un mapa gráfico y editar el trabajo final. Esto serviría, por ejemplo, para poner nuevos elementos, como podrían ser una especie de falda, o unos cuernos o un rabo, a nuestro personaje.

Y ya que lo hemos nombrado, pasemos a ver el editor de mapas gráficos. Aunque parezcan pocas las mejoras hechas en el mismo, existen otro tipo de novedades que, aunque no son parte integrante del mismo, es decir una herramienta dentro de la barra del editor de mapas gráficos, tienen que ver con los mapas gráficos de una manera u otra.

### Editor de mapas gráficos

Algo muy útil y novedoso que incorpora el editor de mapas es la opción de tener una segunda capa tileada, y poder pintar con ella, con cualquier otra herramienta, en el mapa gráfico. Con esta herramienta se pueden usar un par de trucos. Uno de ellos es el de los ladrillos, que consiste en tener un mapa gráfico con un ladrillo y usarlo de textura de segundo plano para ir pintando la




pared de nuestra casa, utilizando por ejemplo el bote de pintura. También podremos dibujarnos guías, teniendo un mapa del tamaño que deseemos con dos rayas pintadas en dos lados formando una ele. Por ultimo, decir que también se pueden crear mapas importables a un FPG utilizando esta herramienta. Para ello, sólo debemos tener la casilla del tamaño del gráfico en otro mapa gráfico en el entorno y utilizarla para construir las demás.

Otra novedad muy útil es el pincel cambiante, del que disponemos en una gran variedad y, si aún así se nos quedan cortos, siempre podremos añadir algunos. Para ello se debe editar el fichero .fpg del directorio system que los contiene. Podremos crear otros y añadirlos pudiendo tener uno con nuestro logo, algo que puede resultar útil.

Se ha comentado ya varias veces las posibilidad de exportar e importar mapas desde y hacia un fichero FPG. Aunque estas opciones están dentro del menú de ficheros, los comentaremos en esta sección. Cuando se exporta un mapa no existe ningún problema, porque es DIV quien lo genera. Pero si queremos importarlo y crearlo nosotros se deben seguir una par de normas. Se debe encuadrar cada gráfico independiente con un cuadrado de un color, por ejemplo el blanco. Luego se debe dejar una separación de al menos un pixel entre cada uno de los cuadrados. Podemos ver un ejemplo en un FPG exportado por nosotros.

## Editor de mapas 3D

La siguiente novedad es el editor de mapas 3D, del que hablaremos a continuación. Éste puede resultar algo complicado, ya que no se parece a otros editores, porque al intentar hacer un editor más funcional se han perdido algunas de las opciones de éstos.

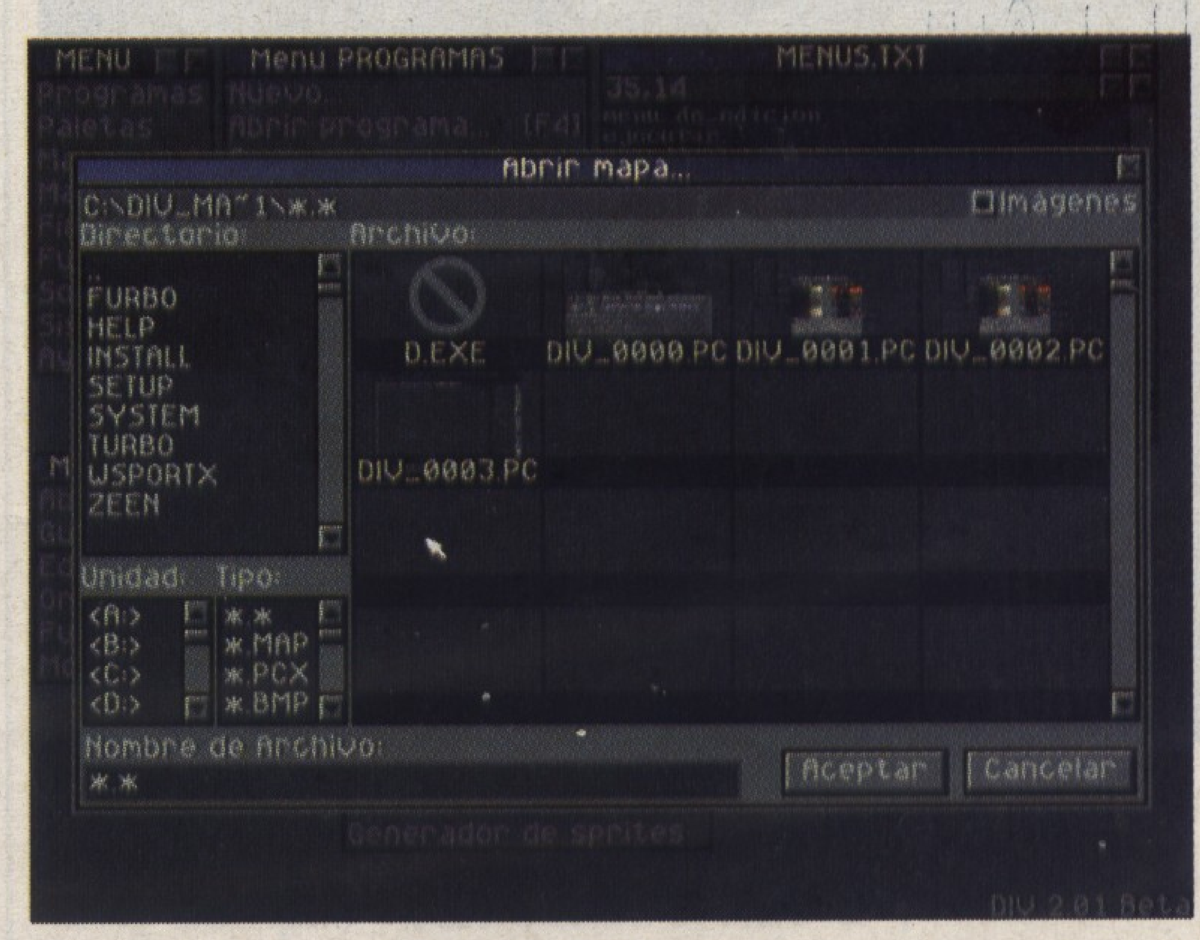
Para no complicar mucho el asunto, iremos poco a poco, explicando cada uno de los pasos que se necesitan realizar para crear un mapa 3D. Empezaremos con la habitación base; pero, antes de realizarla, se debe tener elegido el mapa de texturas que vamos a utilizar. Este mapa debe contener mapas gráficos con un tamaño potencia de 2, excepto para el que usemos de cielo.

Antes de ponernos a colocar vértices, se pueden tener seleccionados las alturas del suelo y del techo, y las texturas por defecto para las paredes, techo y suelo. Una vez que se han elegido todos estos parámetros, pondremos los vértices que tendrá nuestra habitación. Podremos seguir diseñando más paredes y habitaciones, que se construirán con los parámetros por defecto.

Otros objetos dentro del mapa que no tiene que ver nada con las habitaciones son las banderas. Éstas son una especie de puntos de control que indicarán un punto determinado del mapa. Se pueden utilizar, por ejemplo, para colocar al protagonista en el inicio del juego o para crear rutas. Una vez creado nuestro mapa, con sus distintas habitaciones, empezaremos a editarlo. Podemos modificar texturas y vértices y si, por ejemplo, se quiere crear puertas, se debe seleccionar dicha pared y no elegir ninguna textura, por ninguno de sus lados. Otro parámetro que podremos utilizar son las alturas. Con esta opción se podrán crear doseles, ventanas y escalones. Para ello se deben modificar los valores seleccionados para el techo y el suelo. Subiendo y bajando dichos parámetros y creando las habitaciones necesarias, se podrán crear mundos con efectos muy vistosos.

## El lenguaje

Veremos ahora todo lo que tiene que ver con el lenguaje. No veremos ningún caso en particular sino que daremos una especie de pistas a nivel




### Dudas más frecuentes y bugs

Esta claro que cuando aparece una nueva herramienta los usuarios siempre optan más por unas opciones que otras, porque sean más vistosas o atrayentes. En el caso de DIV2, está claro que lo más vistoso son los modos 8 o mundos 3D. Por eso, es normal que este tema haya sido el más comentado entre los usuarios: cómo crear mapas, cómo mover vértices, cómo cambiar texturas y un largo etcetera, relacionado con estos mundos.

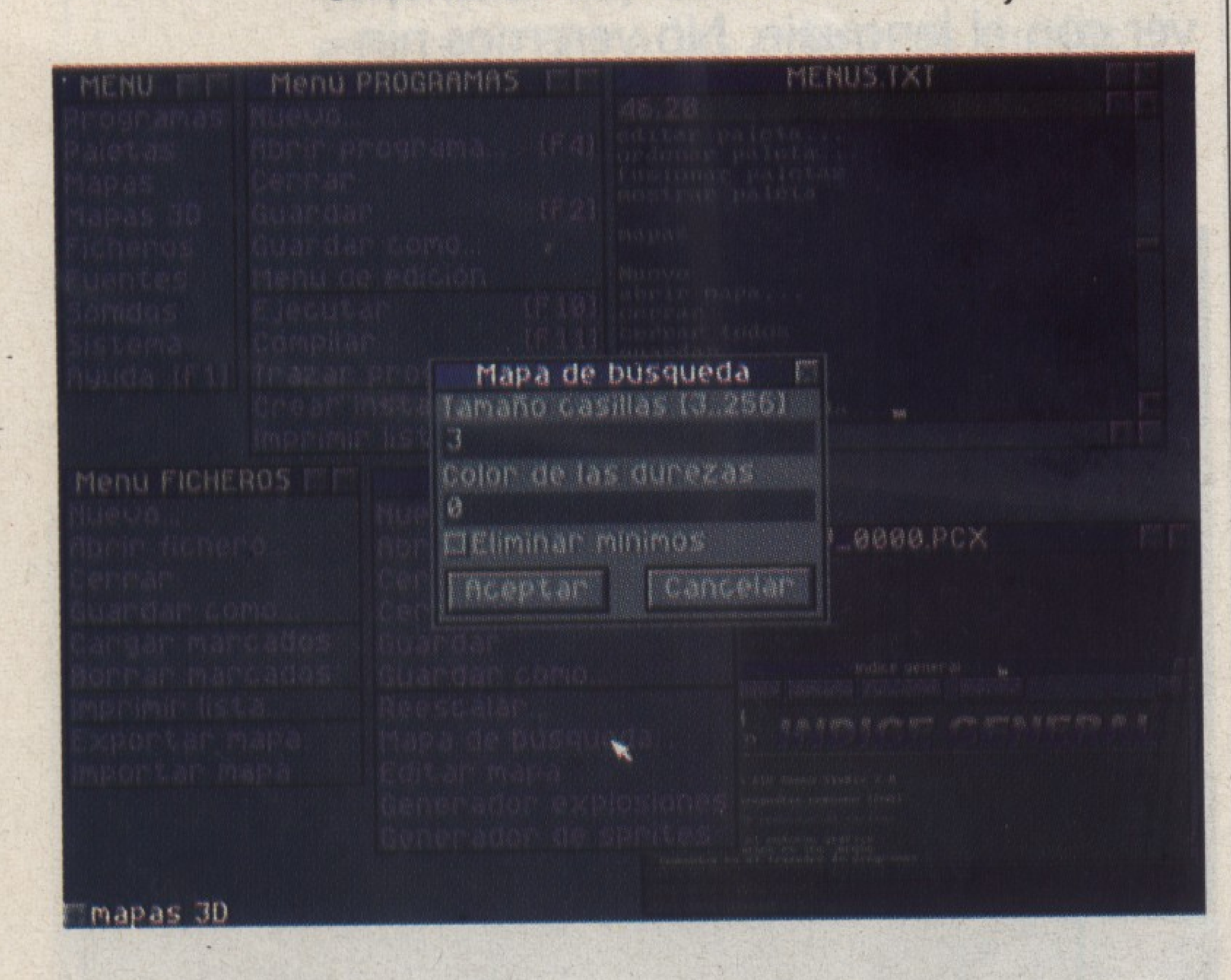
Otro grupo de funciones que ha sido muy preguntado es el de las rutinas de red. Aunque en este caso ha sido algo menos intenso debido a que no todo el mundo dispone de una red a mano. Aun así, siempre resulta bastante atrayente poder crear nuestros juegos para que puedan jugar varios jugadores.

Por otro lado, y eso es algo que sucede hasta en las mejores casas, DIV2 tiene algunos bugs. Esperamos la cooperación de todos los usuarios para conseguir una herramienta lo más perfecta posible. Por eso, se va habilitar un e-mail para recoger todas las sugerencias y detección de bugs que encuentren todos los programadores Div. Esperamos vuestros mensajes.

general por cada una de las partes novedosas. Pero empecemos a ver estas partes, sin más preámbulos.

## Las rutinas de red

Lo primero que vamos a ver son los tipos de conexión posibles, detallando físicamente su forma de funcionar. El primero es el denominado conexión por módem, que consiste en conectar dos ordenadores por módem. Para ello se debe introducir el número de teléfono de uno de los jugadores y llamar a su ordenador, que conectara el módem y recibirá la llamada. El segundo tipo de conexión es el denominado de cable en serie, que consiste en conectar dos ordenadores por el puerto serie, donde normalmente se conecta la impresora. Este cable también tiene la denominación de cable NULL por su configuración especial, ya que no vale cualquier cable de impresora. Por ultimo esta la conexión en red, usando el protocolo IPX. En cualquiera de los tres casos, siempre habrá que determinar el ordenador que hace de servidor y cuál o cuáles hacen de clientes. Esto al principio puede parecer difícil pero una vez entendida la filosofía de trabajo será



fácil implantar el código necesario que conecte dos o más ordenadores.

Aparte de lo antes comentado, lo más importante en una conexión en red, es lo que en el mundillo se ha denominado como los paquetes. Éstos son distintas estructuras con datos comunes. Es decir, estos datos son compartidos por todos los ordenadores que estén conectados. Hay que comentar que estos datos pueden perderse por la red, es decir, que un ordenador introduzca sus datos en el paquete y los demás ordenadores no lo reciban, perdiéndose el paquete por la red. Este hecho ocurre cada vez menos debido a los mejores equipos y conexiones en red, pero si no es ese vuestro caso, es bueno conocer este impedimento. Aun así, esto es sólo algo que tiene que ver directamente con el hardware del que se disponga y no quita que se puedan realizar juegos para echar partiditas entre varios amigos.

## Listas de correo

Existen una serie de listas de correo que para los usuarios que dispongan de Internet pueden ser muy útiles. Todas ellas están creadas en Onelist pero con distintos nombres.

La primera de ellas es la dedicada al canal IRC de DIV, su dirección o nombre es el siguiente: Canaldiv@onelist.com

Esta lista tiene gran cantidad de usuarios subscritos y tiene un movimiento bastante alto de forma diaria.

Además existe otra lista que ha sido creada especialmente para la nueva herramienta, tiene menos usuarios que la anterior y menos movimiento por lo tanto. Su dirección esta en: Div2@onelist.com

Esperamos la suscripción de todo el que quiera participar en un foro Div. Para suscribirse únicamente hay que ir a la pagina de onelist y elegir suscribirse a una lista, de las anteriormente mencionadas. Para quien no lo sepa la pagina del servidor de listas esta en: http://www.onelist.com

Ah, otra cosilla, en breve se actualizará la página oficial del producto, incluyendo demos, conexiones, etc. Mientras se produce esto, pedimos paciencia a todos los usuarios. Gracias.

## Las rutinas 3D

Si hay algo que llama la atención en la nueva herramienta DIV 2 son los mundos en 3D. Por eso, no es raro que dentro de la poca vida de esta nueva herramienta el mayor numero de dudas haya sido sobre lo que se ha dado en llamar modos 8. Y aunque anteriormente existían los modos 7, la nueva versión contiene bastantes diferencias con su predecesora. Lo más importante y novedoso no es otra cosa que las paredes. Anteriormente, únicamente se disponía de suelo y techo, pero con esta nueva versión además se disponen de paredes que podemos decorar con bonitas texturas. También es posible cambiar la textura de dichas paredes, así como del suelo y el techo de cualquier habitación.

Y es que, al igual que el editor de mapas 3D, a la hora de programar todo estará basado en habitaciones. Podremos modificar, además de la textura, la altura del suelo o del techo de cualquier habitación; con esto conseguiremos ascensores y puertas corredizas. Para cambiar de piso se puede usar una técnica de portales, que consiste en teletrasportar al jugador en un punto determinado a otro lugar idéntico pero en otro nivel. Para facilitar la programación de estos portales, se pueden usar las banderas de control.

## Buscando caminos

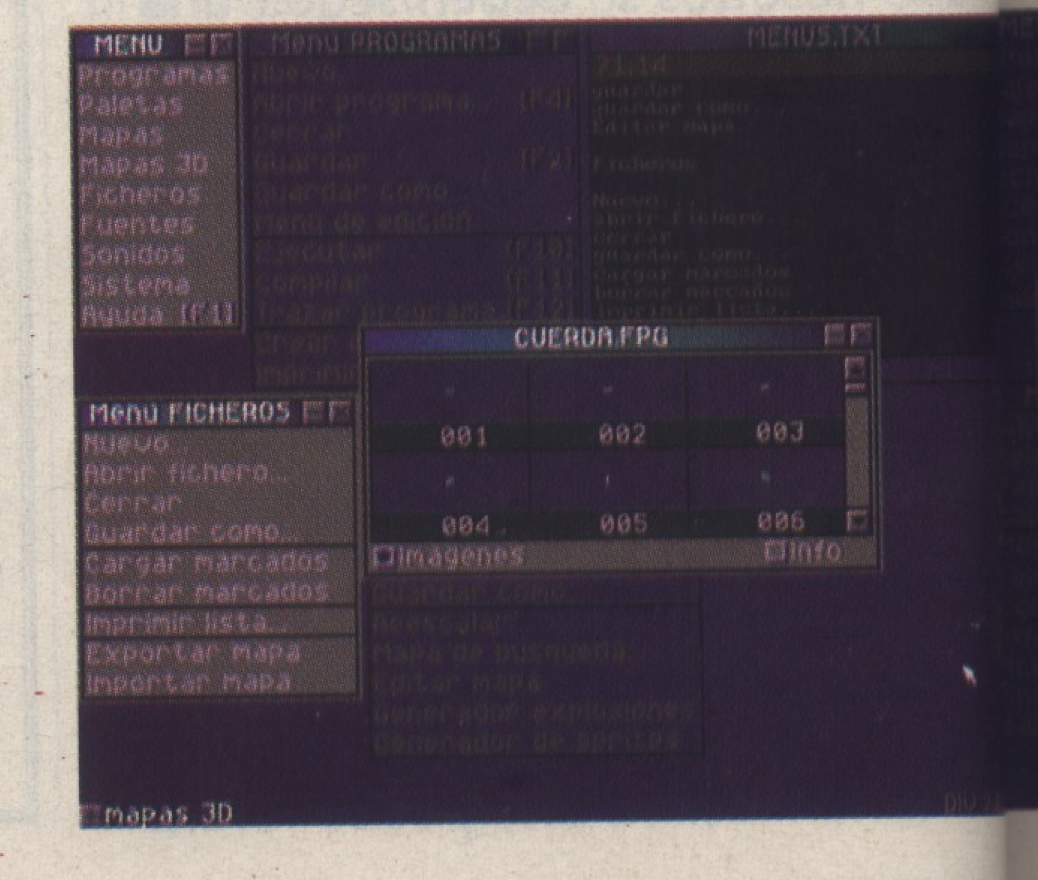
Al igual que el apartado anterior era el más vistoso, el de la búsqueda de caminos es uno de los que se puede sacar más rentabilidad. Un buen juego de estrategia no sería tal si no dispusiese de unas buenas rutinas de inteligencia artificial. Gracias a las nuevas rutinas que incorpora DIV2 nos podremos ahorrar mucho trabajo. Aunque a primera vista dichas rutinas no parezcan útiles, poder mover un personaje rebasando obstáculos entre dos puntos se puede dar en tantos casos en cualquier juego, que cambiaremos de opinión.

Para poder usar dichas rutinas se debe crear un mapa de búsqueda con el generador que incorpora el entorno. Luego se mirarán las posiciones de inicio y final en dicho mapa y se obtendrá una estructura con una ruta para llegar a dicho punto rebasando todos los obstáculos. Las rutinas de búsqueda son bastante rápidas, aun en casos extremos, y podremos ver a nuestros inteligentes personajes moverse con soltura.

## Las rutinas sobre texto

Algo que se podía echar de menos para algunas aplicaciones eran las rutinas de texto, tales como el denominado *input*. Esta función consiste en recibir en una variable un dato desde teclado. Aunque Div2 no posee un *input* como tal, sí dispone de las funciones necesarias para conseguirlo con facilidad. Para realizarlo debemos ir construyendo la cadena poco a poco con las funciones de texto. Por ejemplo, se podrá recoger el dato del teclado con la variable ASCII, e ir construyendo otra variable del nuevo tipo *char.*

Si lo que queremos es introducir un dato numérico, habrá funciones de conversión de datos. Esto también será muy útil para construir fra-



ses, incluso mixtas, es decir con datos numéricos y alfanuméricos. Por ejemplo, se podrá poner desde ahora un cartel del tipo AVidas:3@, que antes era más difícil de conseguir. A la hora de manejar ficheros, si se utiliza un método de construcción de texto para indicar el nombre del fichero, y estos tienen que ir en la instalación, los archivos deben aparecer completos en algún lugar del listado, para que el instalador los tenga en cuenta.

## Rutinas de ficheros

Se han incorporado algunas rutinas de ficheros que pueden ser bastante útiles dado el caso. Con ellas se puede programar un cargador de ficheros automático, es decir, el típico *file manager* de Windows. Una aplicación que permita navegar por los directorios y cargar y manipular cualquier tipo de ficheros. Habrá algún tipo de ficheros como los pcx, map, fpg, wav y pcm que los manejará automáticamente el programa. Pero si vamos a manejar ficheros propios hay que tener algún aspecto en cuenta.

El más importante de todos ellos, y que puede crear dolores de cabeza, es el formato de los datos guardados. Aunque en esta versión existen varios tipos de datos, todos ellos son guardados en memoria en el mismo formato. Éste no es otro que un int o entero de 4 bytes o ,lo que es lo mismo, 32 bits. Esto también se da en el manejo de ficheros, es decir que si guardamos un texto siempre tendrá un numero de letras múltiplo de 4. Esto es así porque cada letra ocupa 1 byte y al final de dicho texto, para rellenar el hueco, se grabarán valores nulo. Hay que tener este aspecto en cuenta tanto al cargar como al grabar cualquier fichero.

## Funciones matemáticas

Aunque anteriormente se disponían de dos funciones seudomatemáticas, que no eran otras que get_ditstx y get_disty, ahora se han incorporado un buen surtido de las mismas. Ya habrá poco cálculos que no se puedan realizar con estas nuevas funciones. Aun así, habrá que tener en cuenta el tipo de valor que devolverá cada función. Normalmente, al ser los resultados un tanto pequeños, se suele dar en milésimas. Esto es así porque en Div no hay números decimales, pero siempre se puede trabajar en milésimas y cuando sea necesario un valor entero real, se dividirá por mil.

Si juntamos estas funciones con las de texto, se podría hacer todo un creador de formulas con DIV2 y representarlo de una manera multimedia. Luego se podría grabar a disco los resultados y formulas con las rutinas de ficheros. Y si fuéramos algo más allá, podría representar musicalmente las formulas. (¡que flipe!) ¿no?

## El sonido y la música

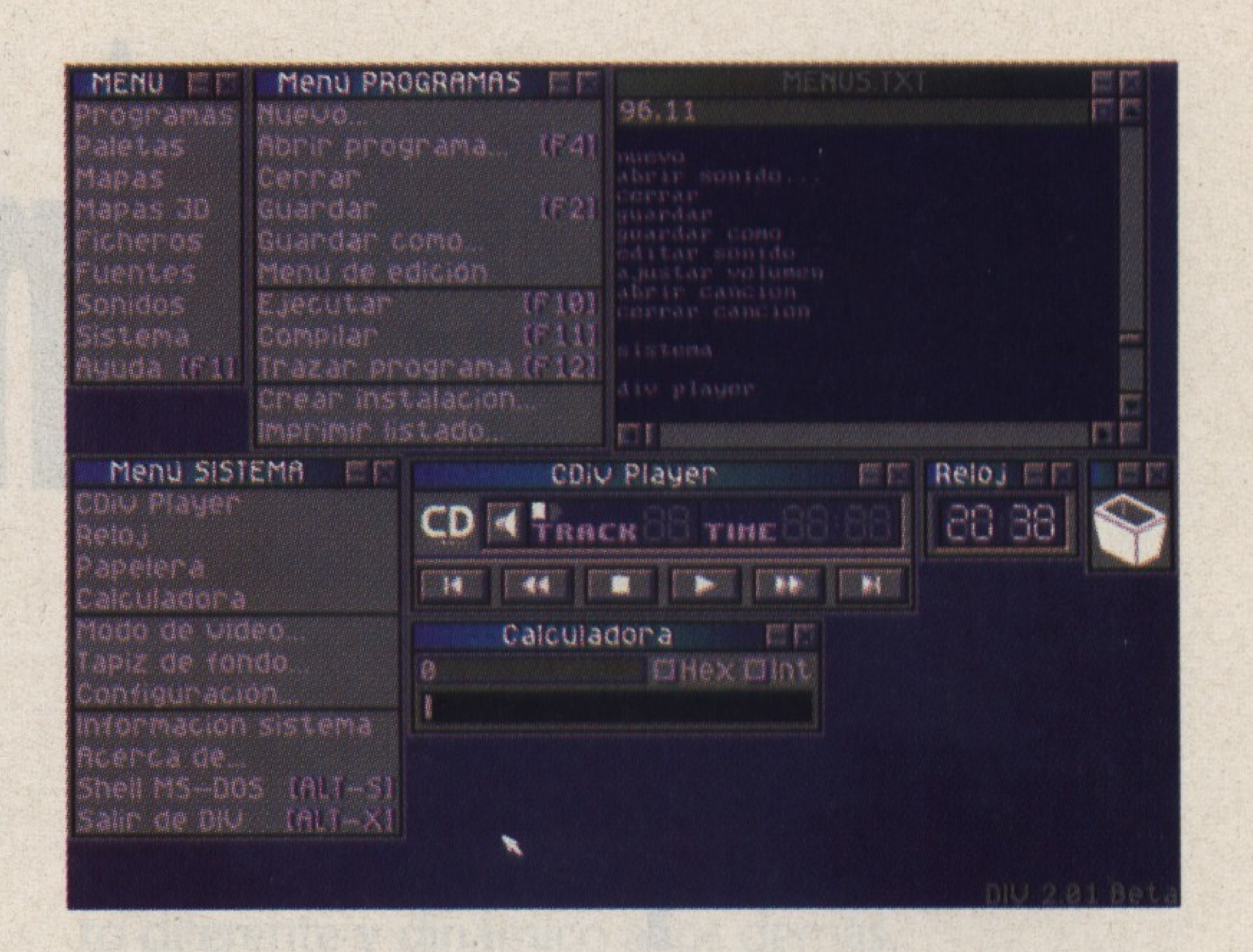
Si bien en la antigua versión ya podíamos dar sonido a nuestras creaciones, en esta nueva versión este aspecto ha sido muy mejorado. Antes sólo se disponía de samples tipo PCM de baja calidad, y de pistas de CD. En esta nueva versión, además de salvaguardar estas opciones, se han incorporado otras como la posibilidad de reproducir ficheros tipo MOD, S3M o XM. Además, se ha mejorado la calidad de los samples; si antes únicamente era posible un formato con una determinada frecuencia, ahora se admiten todo tipo de frecuencias.

Debido a este nuevo tipo de ficheros hay que tener en cuenta el número de canales ya que el tipo de ficheros MOD,S3M o XM pueden usar un numero variable de canales. DIV2 únicamente dispone de 32 canales por lo que hay que tener especial cuidado en verificar el hecho de que nos queden canales para reproducir otro tipo de sonidos del tipo WAV. Lo más recomendable en estos casos es usar canciones de 16 canales y dejar los otros para el otro tipo de ficheros.

Y hablando de ficheros WAV y PCM, hay que tener en cuenta los volúmenes. Debido a que en la anterior versión los volúmenes estaban algo altos, en ésta se han bajado notablemente. Es por esto que en los samples que grabemos deberemos subirlos a un nivel superior para que todo suene con un volumen aceptable.

## Primitivas gráficas

Algo que se echaba en falta en la anterior versión de DIV, y de lo que se hizo incluso una DLL, son las primitivas gráficas. Éstas consisten en dibujos vectoriales como cuadrados, círculos o líneas. En Div2 se han incorporado estas opciones funcionando de forma parecida a los textos, es decir, se colocan en




pantalla y están en ella hasta que se eliminan. Al funcionar igual que los textos, hay que tener cuidado en no llamar a demasiadas primitivas en pantalla para así no saturar al programa. Estas funciones son muy útiles, por ejemplo, para hacer selecciones con el ratón.

## Funciones de memoria

La gestión de memoria se podría considerar programación de alto nivel. Para muchos usuarios este tema no tendrá mucha complicación, ya que llegaran de otros lenguajes que disponen de este tipo de gestión de memoria con punteros. Para los nuevos usuarios, el uso de estas funciones tampoco tendrá mucha dificultad gracias a su simplicidad. En cualquier caso, hay que tener en cuenta que DIV siempre trabaja con datos de 32 bits. Por lo tanto, siempre manejaremos bloques de memoria que serán múltiplo de 4 en su numero de bytes. Además, únicamente se podrán tener 256 bloques de memoria definidos.

## Despedida y cierre

Bueno, con esto quedan vistos casi todos los grupos de nuevas funciones. Hemos comentado casi todas las dudas que han aparecido en la lista, siempre vistas desde la nueva herramienta DIV2. En próximos artículos iremos examinando más de cada uno de los grupos comentados, además de hacerlo de forma más exhaustiva. En cada uno de ellos se propondrá un ejemplo explicativo.

Por último, deciros que esperamos la colaboración por vuestra parte, ya sea proponiendo nuevas dudas o temas a tratar en esta sección. Eso si, siempre tendrá que ver con la nueva herramienta, ya que dejamos la antigua para otras secciones de esta revista. Esperamos que este articulo os haya sido útil; aunque no hayamos comentado ningún tema de forma concreta, hemos intentado dar una serie de pistas generales por cada uno de los temas. Hasta la próxima, Div-eros.

**Antonio Marchal**

# Introducción al diseño gráfico

## Primeros pasos

**En este número vamos a meternos de lleno con 3DS MAX, conociendo cómo nos va a permitir crear nuestros diseños tridimensionales gracias a sus posibilidades.**

En el articulo anterior se mencionaron algunas cosas de las paletas en DIV y la introducción a las 3 dimensiones mediante 3DS MAX. En este articulo se va a tratar más a fondo el 3DS MAX y se explicará cómo hacer algunos efectos especiales con el mismo. También se tratará un poco el Adobe Photoshop.

### Explorando 3DS MAX

3DS MAX es un programa bastante extenso aunque bastante fácil de utilizar si se compara con otros programas de diseño 3D como el Alias MAYA o el SoftImage. La ventaja que tiene 3DS MAX es su calidad/precio en el que no tiene competidores.

Con MAX obtener buenos resultados en un periodo de tiempo relativamente corto es bastante fácil, sólo hay que trabajárselo. La figura 1 es un buen ejemplo de imagen generada con MAX que se ha generado en algo menos de una hora y media.



**Figura 1 - Imagen generada con 3ds MAX.**

Aunque se dispone de primitivas, superficies NURBS y otras herramientas de modelado no se deben olvidar los modificadores que juegan un papel muy importante en este programa. Con los modificadores se puede desde arrugar una superficie hasta suavizarla.

Se va a poner un ejemplo practico de modificador que además servirá para crear un titulo de un juego. Se va a suponer que el juego se llama "80x86 WAR". Lo primero que se debe hacer es entrar a 3ds MAX y pulsar sobre el apartado crear y pulsar sobre el botón de Shapes. Una vez hecho esto, y estando en el apartado de Splines como muestra la figura 2, se debe pulsar sobre el botón que pone *text.*




**Figura 2 - Splines.**

En el apartado donde pone *text* se debe introducir el texto deseado. En nuestro caso, se deberá poner 80x86 WAR. Una vez escrito se debe hacer un clic en la ventana *front* para que el texto aparezca en 3DS MAX. Hasta ahora se ha creado un texto pero QUE NO ES EN 3D. Para convertir este texto a texto tridimensional lo que se debe hacer es ir al panel de modificadores y pulsar sobre el modificador *extrude* tal y como muestra la figura 3.

Entre los valores de configuración de *extrude* hay uno que es, *amount.* Este valor indica la profundidad que se le va a dar a la figura *spline* en sí. Recomiendo ir cambiando el valor y quedarse con el resultado que más te guste. Una vez realizado esto ya se tiene el texto en 3D, pero ahora se debe colocar correctamente en el espacio para hacer un render que quede bien.

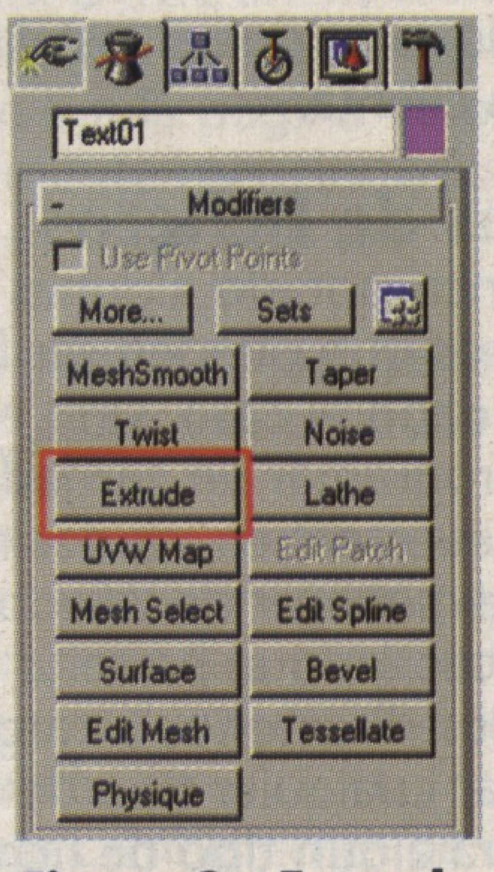



**Figura 3 - Extrude.**

Para ello se utilizaran los botones que hay abajo y la derecha de la pantalla como los que muestra la figura 4.



**Figura 4 - Botones para controlar la vista.**

A continuación se va a describir la función de cada uno de estos botones empezando por los de arriba de izquierda a derecha y después los de abajo con el mismo orden.

El primer botón sirve para acercar o alejar la vista a una zona. Funciona como la lupa de cualquier programa de dibujo o diseño. El segundo botón es básicamente lo mismo pero lo que le diferencia del botón que tiene a su izquierda es que ése sólo afecta a la ventana en la que se está modificando mientras que este segundo botón afecta a todas las ventanas. El tercer botón tiene una utilidad muy especial, esta utilidad es la de hacer que la vista se ajuste a los objetos que hay en la escena de manera que si se tiene una ciudad hecha en 3d y se pulsa sobre este botón, la vista se pondría lejos de manera que pudiera ver toda la ciudad.

Los botones inferiores también son muy útiles. El primero sirve para acercarse o alejarse igual que el que tiene una lupa pero la diferencia reside en que cuando te alejas o te acer-

cas usando este botón la perspectiva queda cambiada. Para entender esta diferencia, se pueden observar las figuras 5 y 6.

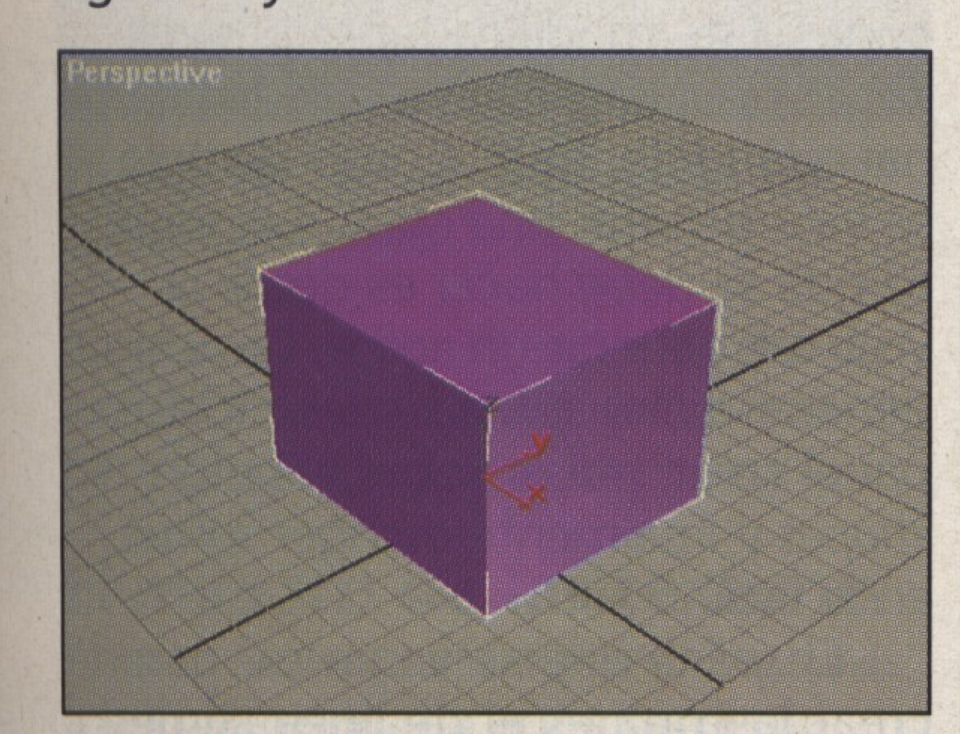



**Figura 5 - Cubo usando la lupa.**

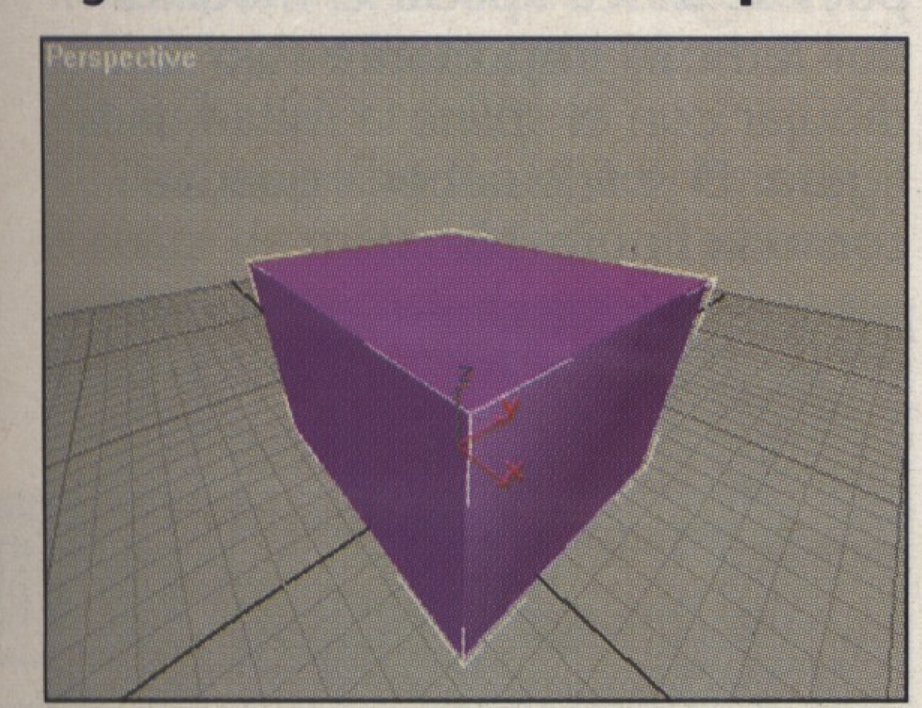



**Figura 6 - Cubo distorsionado.**

El segundo botón sirve para desplazarse arriba, abajo, a la derecha o a la izquierda pero nunca rotando la perspectiva, con lo que se puede ver qué hay a los lados de un objeto. El tercer botón sirve para exactamente lo contrario, con este botón se puede rotar la perspectiva pero no te mueves arriba, abajo, a la derecha o a la izquierda. Finalmente, el tercer botón de abajo sirve para hacer que la ventana activa ocupe toda la pantalla para verla con más detalle y poder añadir cosas, o simplemente verla mejor.

Recomiendo examinar e ir probando estos botones por cuenta propia, es la mejor forma de aprender a dominarlos una vez explicada la teórica.

Volviendo al ejemplo de las letras, ya se tienen los conocimientos básicos para rotar y colocar el objeto tridimensional correctamente en el espacio.

Una vez colocado el objeto en el lugar deseado, lo único que se debe hacer es el render, proceso que ya se explicó en el articulo anterior pero que se recordará brevemente en éste. Para acceder a la ventana de render, lo único que se debe hacer es ir al menú rendering y pulsar en render. Con esto se abrirá una ventana de configuración como la de la figura 7.

En el apartado de *output size* se debe indicar el tamaño de la imagen en pixels. Puede ser 320x200, 640x480, 800x600... Si te fijas

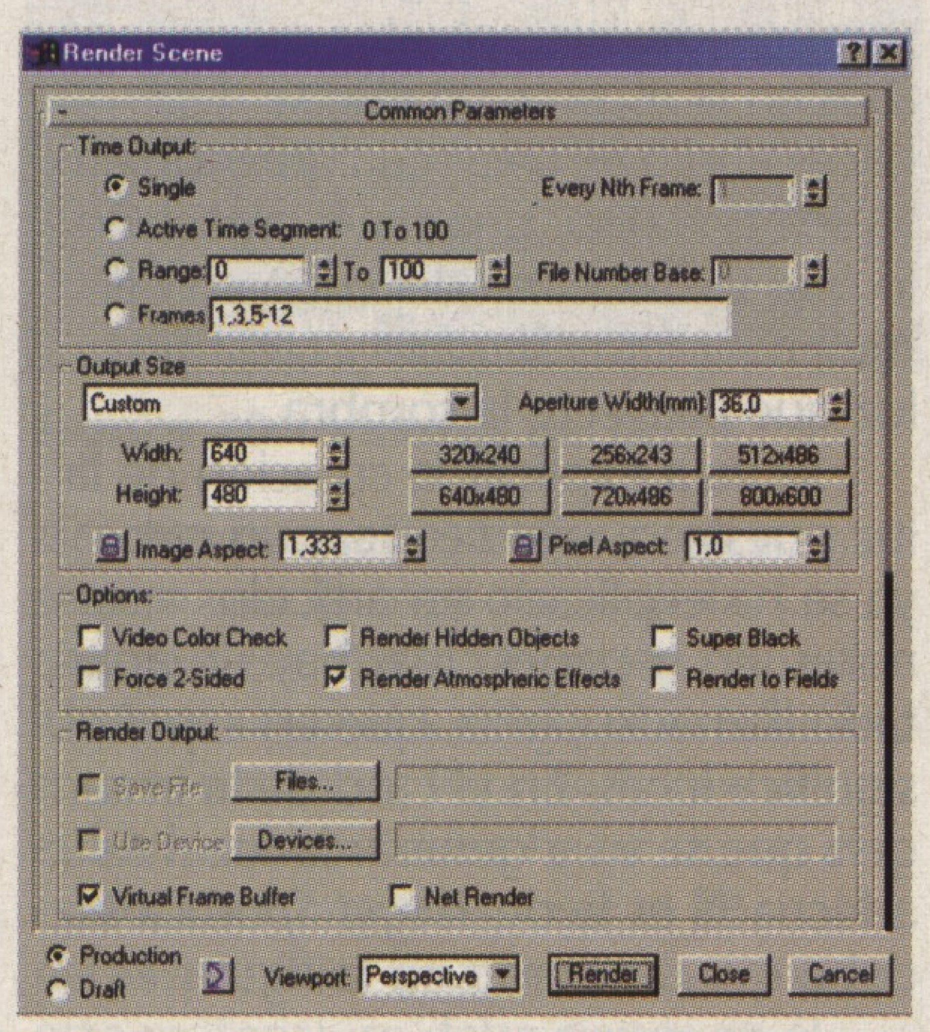



**Figura 7 - Ventana de Render.**

éstos suelen ser los tamaños que se usan para los modos de pantalla que se utilizan en los juegos. Esto es ideal en el aspecto que podemos hacer gráficos en 3ds MAX para los juegos que encajarán con el tamaño de la pantalla. Sólo hace falta diseñarlos al tamaño correspondiente.

En la ventana de render también es importante remarcar que en el apartado *output render* hay una opción llamada *save file;* esta opción es la más importante puesto que sirve para grabar la imagen en un fichero. Para hacerlo se debe pulsar sobre *files* y escoger el fichero en el que se quiere guardar la imagen.

Ahora que se sabe todo esto se esta preparado para hacer el render. En el caso del autor de este articulo, el render ha quedado como el de la figura 8 pero el resultado depende de cómo se haya distribuido en el espacio este objeto.

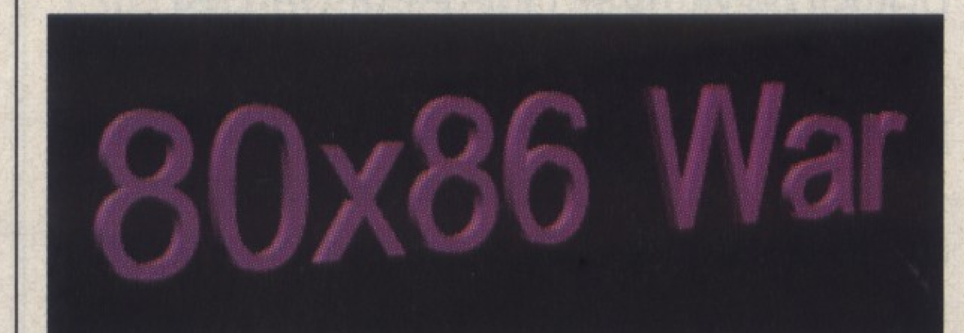



**Figura 8 - Resultado del Render.**

Con unos pocos efectos se puede mejorar mucho una imagen que en principio es bastante "sosa", como la que se ha generado hasta ahora. Un efecto muy interesante es el de *volume fog.* Este efecto se puede activar desde *rendering/environment.* Primero se debe pulsar sobre ADD y pulsar sobre *environment fog.* En la configuración de este efecto, hay una opción llamada *density.* Modificando esta opción haremos que la niebla sea más o menos densa. Aplicándolo como es debido se puede conseguir un efecto como el de la figura 9.

Estas imágenes pueden ser útiles para la pantalla principal de un juego

80x86 War

**Figura 9 - Resultado del Render con Niebla.**

o para una región del mismo. Por ejemplo, en un juego de naves es interesante hacer un panel a la izquierda o a la derecha con imágenes 3D. Esto le da al juego un aspecto diferente y dinámico a los demás.

En el apartado de *rendering/ environment* también hay un efecto llamado *fog* que en español significa niebla. Este efecto también es muy útil y da a la imagen un aspecto realmente impactante. En la figura 10 se puede observar una secuencia de imágenes generadas con 3ds MAX y el efecto de la niebla.

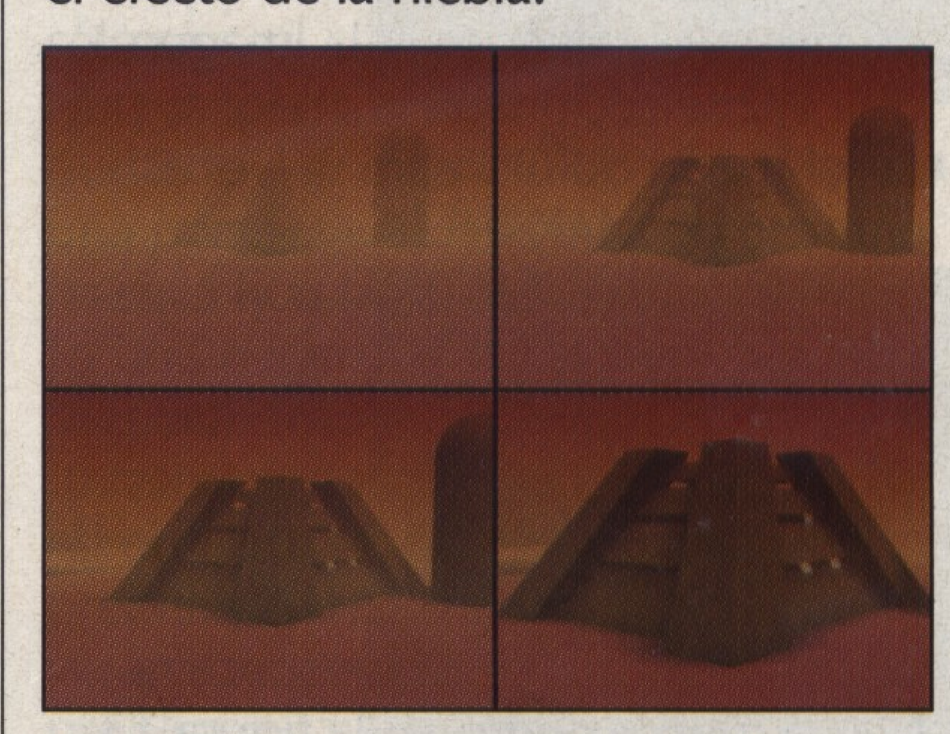


**Figura 10 - Secuencia de imágenes.**

Para conseguir buenas imágenes a veces es necesario recurrir a más de un programa para que el resultado final sea excelente. Por ejemplo, aunque desde 3DS MAX se pueden incluir destellos de luz, a veces es recomendable hacerlo desde otros programas como el Adobe Photoshop. Un ejemplo de esto se puede observar en la figura 11 en la que hay la imagen generada con 3ds MAX y el destello realizado con Adobe Photoshop.

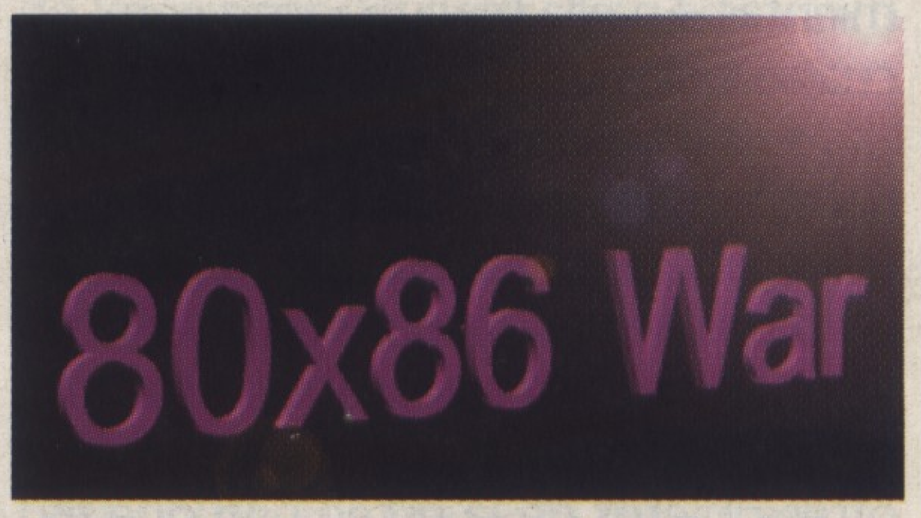
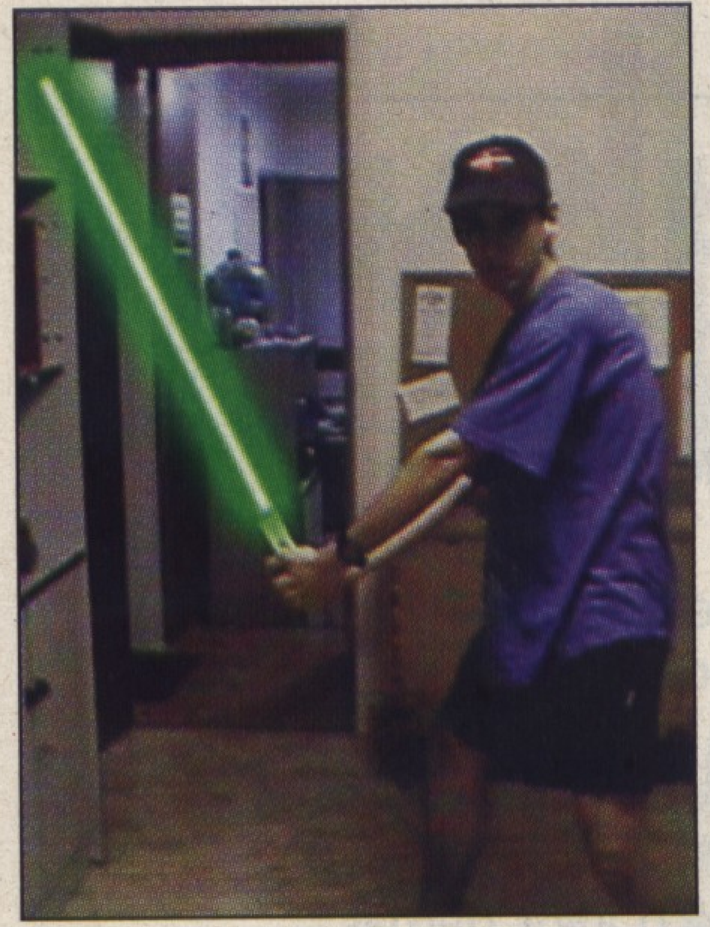



**Figura 11 - Imagen con destello de luz.**

Tenemos que recordar que Adobe Photoshop es un programa de retoque fotográfico profesional con el que se puede hacer cualquier cosa. Desde hacer que una foto antigua se vea mas nueva a convertir a chicos en guerreros JEDI como es el caso del chico de la figura 12.

El proceso de convertir a este chico en JEDI es uno de los más sencillos, aunque hay que conocer bien la herramienta a utilizar para poder hacerlo.


**Figura 12 - Un chico convertido en JEDI.**

## Fusionando la realidad y la ficción

Algo que también es muy interesante es la inclusión de objetos tridimensionales en fotografías que como todos sabemos son en 2D. Ahora se va a explicar como poner un objeto 3D en una fotografía. Lo primero que se debe hacer es escanear la fotografía o coger una existente del disco duro. Después se debe entrar en 3ds MAX y seguir los siguientes pasos:

- Se debe seleccionar la vista perspectiva e ir al menú *views*. Una vez allí pulsar sobre *background image*.
- Una vez dentro, usar el botón de *files* para seleccionar la imagen. Una vez seleccionada la imagen, pulsar el botón de OK para volver al entorno de MAX.
- Ahora se debe ajustar el espacio tridimensional a la imagen con las herramientas antes comentadas. Lo que se quiere decir con esto es que se debe hacer que el suelo de la fotografía y la rejilla de MAX parezcan estar en la misma perspectiva. Mas o menos como en la figura 13.
- A continuación se debe crear un cubo, preferiblemente desde la vista *top*.


**Figura 13 - ajustando la rejilla.**

- Una vez creado el cubo se debe ir al editor de materiales. Para acceder a él se debe usar la barra de herramientas. En ella hay un icono con cuatro bolas de diferentes colores.
- Una vez en el editor de materiales se debe cambiar el tipo de material que por defecto está a *standard* por *Matte/Shadow*, que es un tipo de material transparente pero que es capaz de recibir sombras de otros objetos de la escena, lo que es ideal para intercalar imágenes reales con imágenes sintéticas.
- En la configuración de este tipo de material se puede observar un apartado llamado *shadow*. En él hay la opción de activar la opción de *recibe shadows*. Para realizar la imagen que hemos planteado es necesario activar esa casilla, lo que significa que nuestro objeto será transparente pero recibirá las sombras de otros objetos.

Hasta ahora lo que se tiene es un objeto transparente que va hacer que nuestro campo de trigo reciba las sombras de otros objetos que pongamos en la escena. Lo que se debe hacer ahora es crear cualquier objeto, ya sea un cubo, una nave espacial o lo que sea, encima del cubo con el material *Matte/Shadow* y crear una luz que proyecte la sombra.

Se va a explicar la creación de la luz puesto que la de crear un cubo ya se ha explicado en artículos anteriores. En el panel crear hay un apartado llamado *Lights*, que en español significa luces. En el caso de la imagen que se va a generar la luz más adecuada es la de *Target Spot*. La luz debe quedar como en la figura 14.

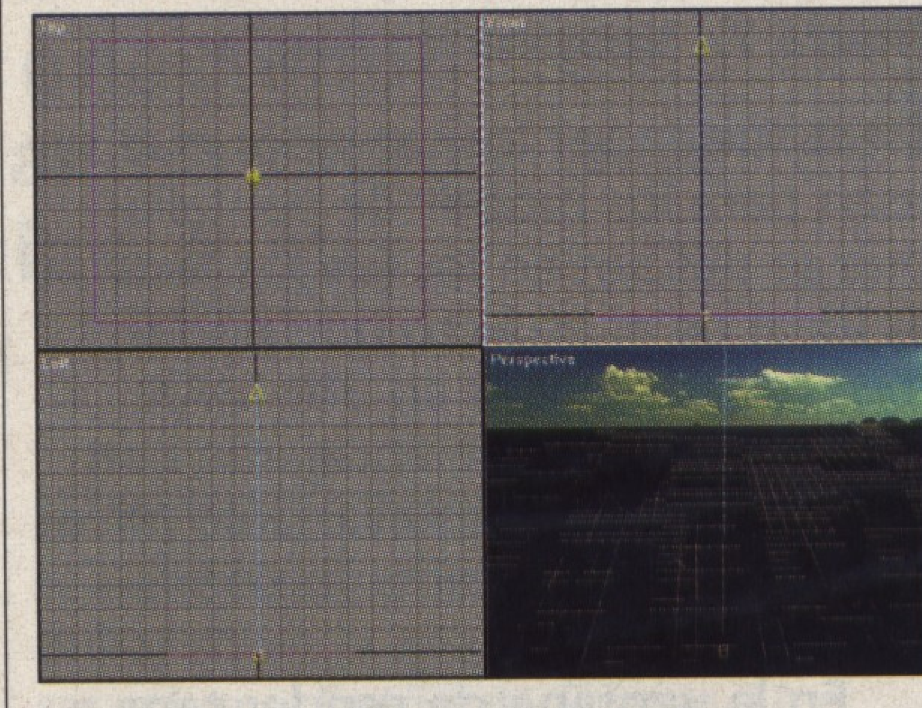

**Figura 14 - Ajustando la luz.**


**Figura 15 - Imagen final.**

Después se debe seleccionar la luz e ir al panel de modificadores. En las propiedades de la luz, hay una opción a activar que es la de *cast shadows* que servirá para que la luz proyecte las sombras de los objetos. Si no se activa esta opción, sólo iluminará a los objetos pero no proyectará las sombras.

Una vez realizado todo este proceso ya se puede hacer el render y observar el resultado final. El autor de este articulo se ha decidido por hacer unos ovnis volando por el cielo mientras proyectan sombras.

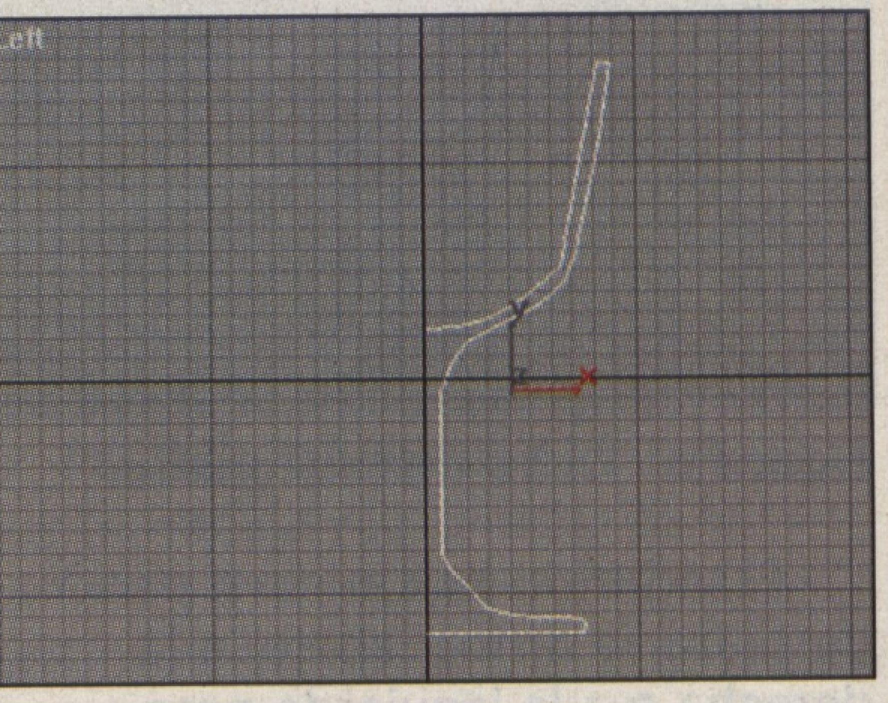
## Modelando en MAX

MAX va mas allá de crear textos tridimensionales o cubos. Es un programa realmente potente capaz de generar cualquier cosa imaginable. Ahora se van a examinar algunas de sus principales funciones de modelado que resolverán algunas dudas que se puedan tener como la de.... ¿cómo se crea una copa ?

Se va a intentar que el proceso de aprendizaje de 3ds MAX sea bastante práctico con lo que se pueda aprender fácil y agradablemente.

Hay un modificador de *splines* muy útil llamado *Lathe* que es con el que se generan objetos como las copas. Este modificador se basa en coger un *spline* y hacerlo tridimensional girándolo 360º. Como ejemplo se va a generar una copa. Lo primero que se debe hacer es desde la vista *left* generar un *spline*. Para generar dicho *spline* se debe ir al panel crear, pulsar sobre el botón *shapes* y finalmente escoger *splines* en la lista.

El *spline* que se va a usar para crear la copa es el de línea. Si después se desea aplicar el modificador *lathe* es importante solo se dibujar con el *spline* la mitad de la figura. El *spline* debería quedar como la figura 16.


**Figura 16 - Perfil del spline.**

Una vez realizado el *spline* se debe aplicar el modificador *lathe*. Para hacerlo se debe ir al panel de modificadores y hacer clic sobre el botón *lathe* ( todo esto con el objeto seleccionado ). Una vez aplicado se deberá probar a pulsar los botones de Min, Max, y Center para escoger el resultado que más convence en el apartado *Align* de las propiedades de *lathe*. Una vez realizado esto, el resultado en la vista perspectiva debería ser algo parecido a la figura 17.

**Figura 17 - Copa final.**

Si se quiere añadir más resolución a la copa, deberemos cambiar el numero que está en la opción *segments*. Contra más alto sea el numero, más resolución tendrá.

Esta escena resultante puede parecer un poco simple y sin vida pero se puede mejorar sin mucho esfuerzo. Utilizando unas buenas texturas y una buena iluminación se pueden llegar a obtener resultados como los de la figura 18. La figura 18 ha sido generada con la misma copa que se ha creado antes pero con una buena textura de cristal. A el suelo que es un simple cubo, se le ha aplicado una textura de madera y









**Figura 18 - Imagen después de aplicar texturas.**

finalmente se han creado un par de luces con la opción de *cast shadows* para que proyectaran sombra.

## El milagro de las texturas

Ahora se va a pasar a poner texturas a la copa realizada anteriormente. También añadiremos un suelo con textura y unas luces para hacer la ilusión más real.

Para empezar se debe entrar en el editor de materiales. En el primer material, debemos desplegar el apartado de *maps*, en él hay un apartado llamado *refraction*. Se debe pulsar en el botón que pone *none* y escoger de la lista el elemento llamado *raytrace*. Una vez realizado esto ya se tiene una textura de cristal pero que es 100% transparente con algunos brillos por lo que aún no es real. Para volver al menú donde teníamos los mapas, se debe pulsar un botón con una flecha como el de la figura 19.



**Figura 19 - Volviendo a lo anterior...**

Si se observa con atención se puede ver que al lado de donde pone *refraction* hay un valor que va del 0 al 100. Este valor es el grado con el que se aplicará la refracción que se le dará a la copa. Contra más alto sea el valor, más transparente será la copa, mientras que si lo ponemos muy bajo, la copa parecerá de plástico. Para la imagen anterior, el autor usó el valor 94.

Otro factor importante es el del brillo de la copa. Si uno se fija en la vida real podrá observar que todos los objetos de cristal brillan. Pueden brillar más o menos pero siempre brillan. Así pues se debe hacer que el material realizado con 3ds MAX brille para parecer real. Esto se hace desde el apartado *basic parameters* del material actual. El valor *shinines* es el que indica el grosor del brillo mientras que el valor de *Shin Strenght* sirve para indicar el valor de brillo que tendrá. Para obtener resultados convincentes no hay un valor predefinido puesto que cada tipo de cristal es diferente, pero se recomienda poner un valor alto *Shin Strenght* y un valor algo bajo para el valor *shinines*. Resumiendo, algo como la figura 20.

Posiblemente si ahora se hace un render lo único que se podría ver son

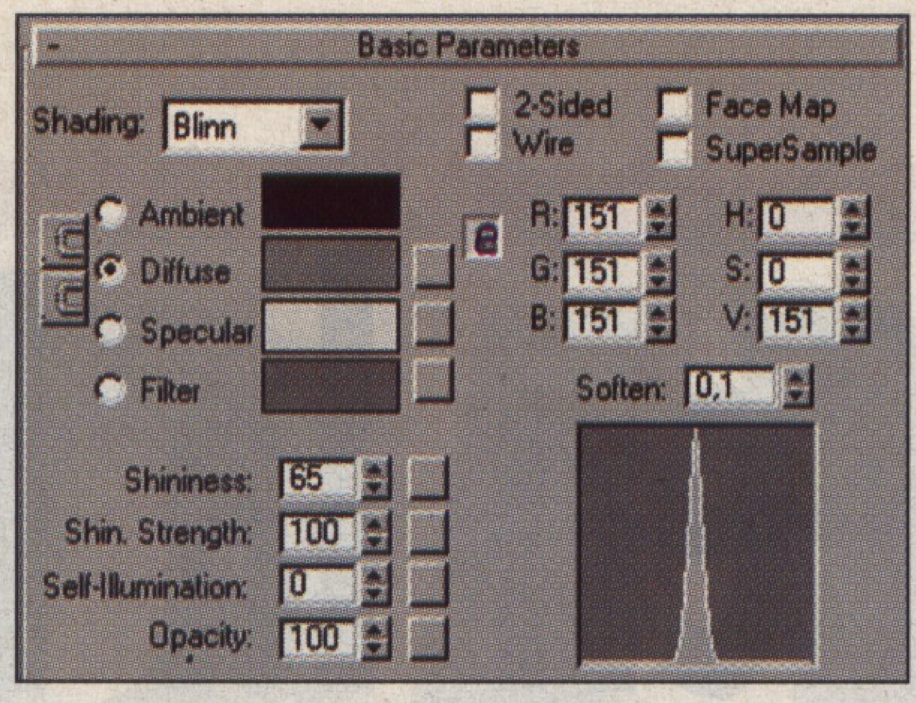



**Figura 20 - Valores de brillo.**

algunos pequeños brillos pero casi seguro que la copa no se vería. Lo que se debe hacer ahora en primer lugar es crear el suelo que es básicamente un cubo. Para darle textura al cubo lo que se debe hacer es, en el editor de materiales, usar el 2º material disponible. Una vez seleccionado este material, se debe poner en el campo *diffuse* del apartado *maps* un *bitmap*. En el caso de la imagen del autor del articulo, se ha usado una textura de madera pero se puede utilizar cualquier tipo de textura.

Si a continuación se hace un render se podrá observar que se ve una copa encima de una mesa pero no es del todo real. La figura 21 ilustra una posibilidad de resultado.

La solución de este problema es la implantación de luces que proyecten sombras en la escena. Para empezar crearemos una luz *Target Spot* ( ya comentada anteriormente en el articulo ) Es importante acordarse de activar *cast shadows* para que proyecte las sombras y conseguir un resultado más real. Después de realizar esta operación ya se está preparado para hacer el render final. El resultado de la escena deberia ser semejante al de la figura 22.

## ¡¡ La refracción es real !!

Se puede observar en la imagen que la copa hace que lo de detrás se vea un poco mas grande y desplazado. Esto es por el efecto de la refracción. Esto significa que MAX nos calcula directamente la refraccion de la copa. Para comprobar que es cierto puedes poner un cilindro entrando en la copa para ver el impactante resultado. Lo mismo pasa si se pone detrás. Recomiendo hacer pruebas para ver los excelentes resultados que puede llegar a ofrecer este sistema.

## Conclusión

Todo esto que se ha enseñado en este articulo es realmente importante ya que si uno se fija en su juego favorito, ya sea en la presentación o en el mismo juego, éstos hacen muy atractivo el mismo. A veces sin conocer un juego, dependiendo de los gráficos, nos interesamos más o menos por el mismo. Lo enseñado en este articulo puede servir principalmente para presentaciones de un juego o para secuencias intermedias en las que se van explicando cosas nuevas. Pero se debe reconocer que con 3ds MAX se pueden hacer gráficos para presentaciones o para el mismo juego. Un ejemplo práctico de esto es que los personajes y mapas del juego *Quake III* están desarrollados con 3ds MAX al igual que los personajes de juegos como *Age of Empires* o *Comandos*.

## En caso de problemas

Si se intenta hacer algo explicado en el articulo y surgen problemas se puede escribir a ksc_dmc@geocities.com indicando el problema y el mayor numero de información de cómo está lo que se está haciendo. Con ello no sólo se resolverá la duda de una persona sino que todas las personas que tengan el mismo problema verán ese problema resuelto en el próximo articulo. También se puede escribir a esa dirección de correo en caso de querer compartir algún truco que se crea que los otros usuarios de programas de diseño y 3D no conocen.

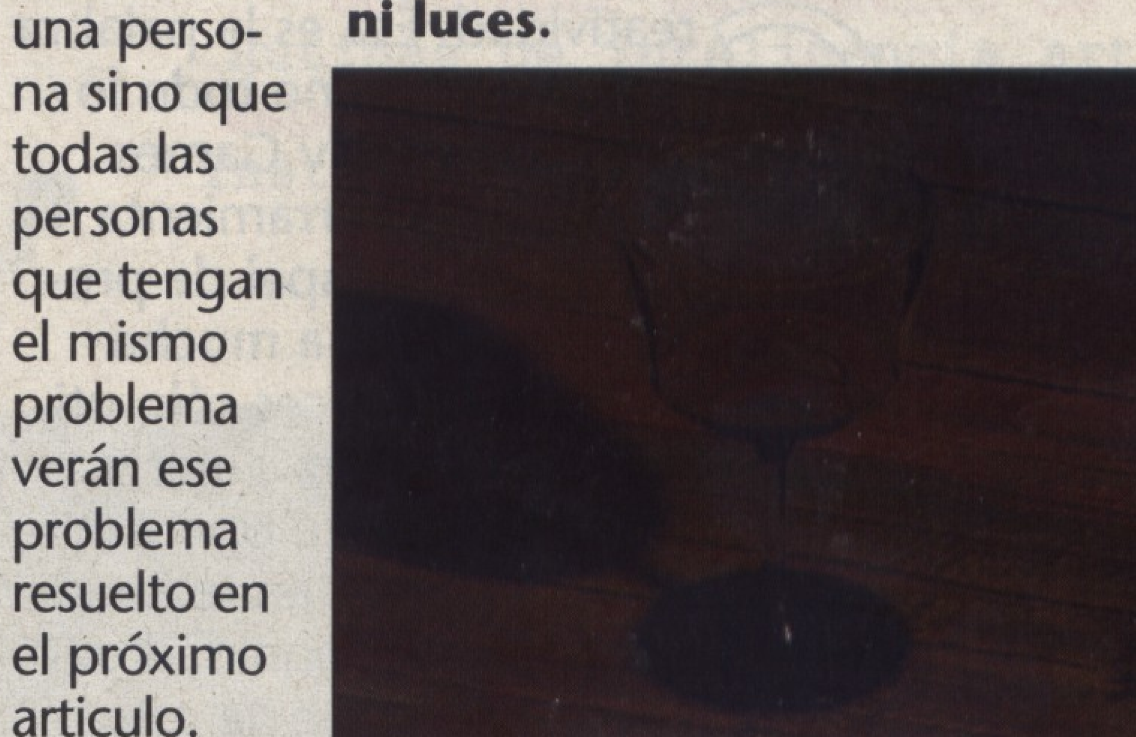


**Figura 21 - Imagen sin sombras ni luces.**

**Figura 22 - Imagen con luces y sombras.**

## En el siguiente articulo

En el siguiente articulo se explicará como hacer gráficos para el interior de los juegos. Es decir, este articulo es más sobre gráficos para presentaciones o secuencias intermedias mientras que en el próximo se tratarán temas como la creación de personajes virtuales sencillos para implantar en los juegos y luego se examinaran las técnicas más idóneas para conseguir que este personaje tenga un aspecto más real. También se tratará el tema de los decorados y se escribirá algo de código en DIV para hacer una especie de juego plataformas. En el próximo articulo también se comentarán las novedades del DIV2 relacionadas con los gráficos, de manera que se pueda empezar a trabajar con él aprovechando ya su máxima potencia.

Por David Martínez (*d_martinez@geocities.com* ) y Alex Huguet



# DIVersión en la Red

## Proyectos y utilidades en Internet

**Muchas páginas se olvidan en el constante fluir de la información en Internet. De hecho, sólo permanecen aquellas cuyo contenido está tan abierto al cambio como la propia Red. Tal es el caso de las webs que tratan de Div: proyectos, ideas, juegos, cursos prácticos toda una riqueza informativa en constante evolución y, lo que es más importante, al alcance del cualquiera.**

Creatividad. Ésa es la palabra que mejor define todo lo que rodea a Div Games Studio, una herramienta que ha puesto en el papel de programadores-creadores a mucha gente que hasta entonces sólo utilizaba el ordenador para jugar. La reacción ante semejante oportunidad no se hizo esperar en la Red. Hoy, ya existe una comunidad virtual afianzada alrededor de este programa, una comunidad abierta en la que prima el intercambio, las propuestas y las nuevas ideas. Mañana, si el software nacional evoluciona como todos esperamos y alguien decide hacer un estudio del porqué de este progreso, su tesis habrá de pasar irremediablemente por Div y, sobre todo, por todos aquellos grupos que hoy hacen de éste un medio por el que canalizar su imaginación.

### Empezamos por...

la página oficial, por supuesto. Aunque hay que tener muy presente que las aportaciones más interesantes (como comentábamos en el anterior número) vienen de la mano de webs particulares, quizá porque se realizan sin ánimo de lucro, con el único fin de ofrecer e intercambiar programas y en el ambiente de cordialidad que existe en Internet cuando se encuentran personas que comparten los mismos intereses. Pero es en la página oficial, aun siendo más fría y previsible, donde llegan las primeras novedades. La lista de *Links* también es un punto a su favor, aunque nosotros nos quedamos con la sección *Downloads* donde podréis encontrar, entre otros, interesantes juegos de usuarios de Div con su correspondiente comentario.

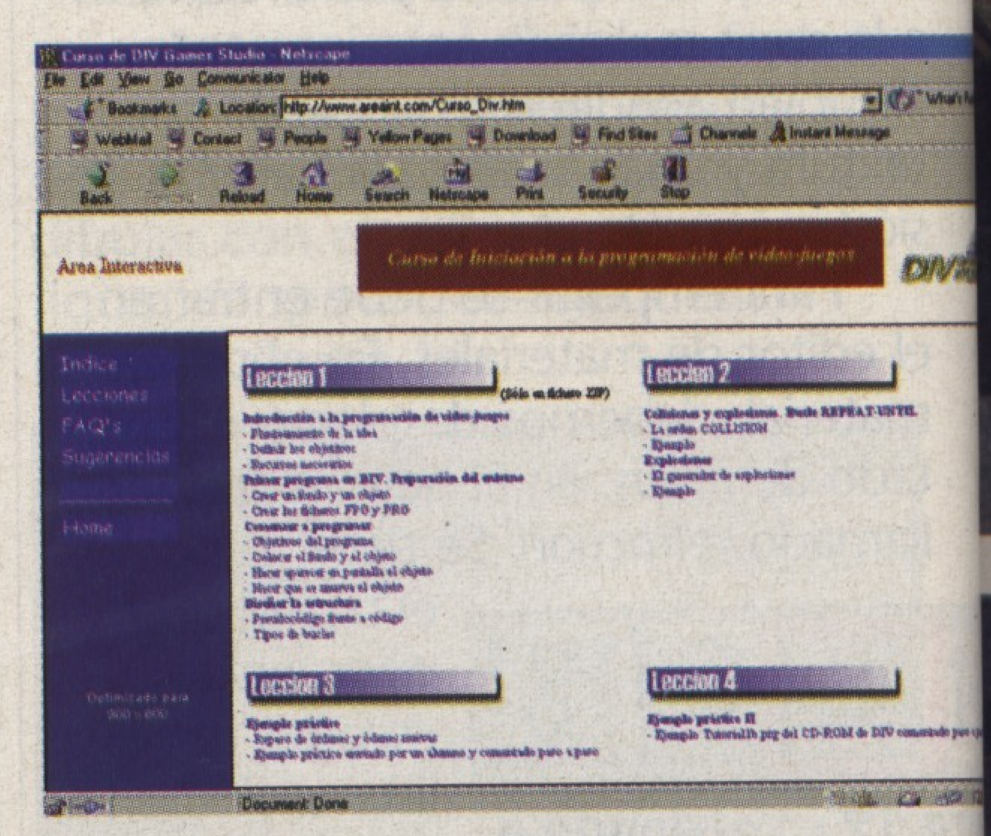


**Pantalla principal de la página oficial de DivGames.**

También, como primer paso, podemos considerar el aprendizaje. Si ya dispones de Div Studio y te has leído el manual, a lo mejor convendría que te pasaras por Área Interactiva, una página dedicada a la formación a través de Internet que incluye cursos de todo tipo, desde la ofimática hasta el diseño, pasando por el perfeccionamiento en el uso de la herramienta que aquí nos ocupa. Un curso muy bien estructurado en cuatro lecciones, esquematizadas y con ejemplos, que puedes encontrar en la sección Especial dentro de los cursos gratis de esta práctica página.

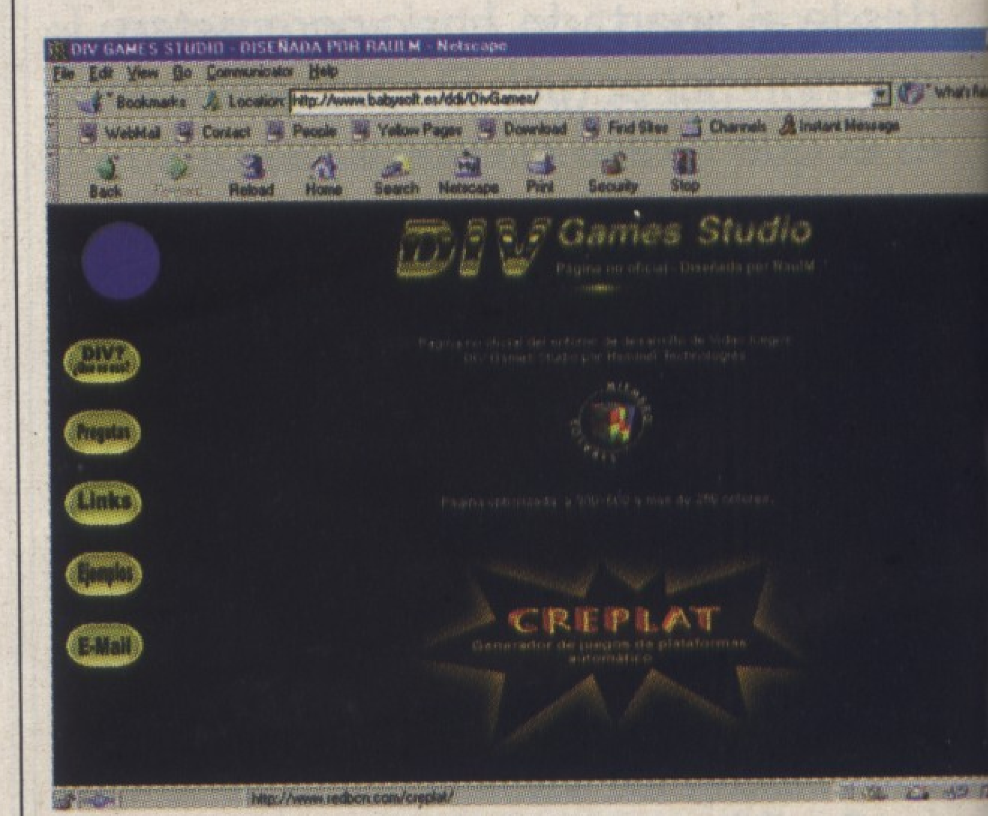
### La excelencia

Después de conocer un poquito cómo está la situación, nos dirigimos a aquellas páginas donde uno se puede abastecer: programas, juegos y alguna que otra curiosidad. Como DDI, una web bien diseñada con un contenido algo ecléctico. En el apartado que más



**Un práctico curso simplificado en 4 lecciones.**

nos interesa (a efectos de esta revista), encontramos un breve comentario sobre Div Games Studio (todavía no se habla por aquí del 2), unos cuantos ejemplos sobre su utilización y algún que otro *link* interesante, sobretodo, para los que busquen contactar con programadores y encontrar oportunidades dentro de este mundillo. Con todo, lo que más nos ha llamado la atención de la página de RaúlM es que aquí nos podemos hacer, previa suscripción gratuita, con un interesante programa denominado *Creplat 0.5*, un generador automático de juegos de plataformas. Con esta interesantísima herramienta no tenemos ninguna necesidad de programar o revisar un código y manipularlo, sino que basta con introducir los gráficos pertinentes y crear la instalación desde DIV Games Studio.




**Si eres amante de las plataformas, hazte con Creplat.**



Juegos comerciales con DIV 2
Editor de sonidos.
Laberintos 3D.
Editor de mapas 3D
Juegos en isométrica.
www.divgames.com
DIV 2
Y porsupuesto, seguiremos demostrando que...
cualquiera puede hacer juegos
¿ Cualquiera ?
Ahora con funciones 3D
Ahora podrás crear juegos en red
Editor de mundos tridimensionales
Browsers para todo tipo de ficheros
Generador automático de sprites
Rutinas de inteligencia artificial
Mapeador de niveles para juegos 3D
Instalador profesional configurable
Más de 1.000 Bugs solucionados
Código optimizado para Pentium
Rutinas para manejo de textos
Sistema de sonido mejorado
Compilador más optimizado
Y un sin fin de funciones
DIV manía
POR FIN LLEGA DIV 2
Revista Oficial
DIV GAMES
De venta en quioscos, grandes almacenes y tiendas especializadas
Teléfono distribuidores +34 91 304 06 22 (ext. 137)
sólo 4.995 ptas.
(bueno... habrá quien todavía no se aclare)
DIV 2
Manual de 348 págs.
LA HERRAMIENTA PERFECTA
Un nuevo entorno de desarrollo que ha evolucionado tanto en sencillez de uso como en potencia y capacidad. Y además se han incluido un gran numero de herramientas nuevas que harán que DIV 2 y tu imaginación formen un equipo perfecto.
COMPATIBILIDAD 100% DIV 1
Esta versión de DIV 2 mantiene compatibilidad al 100% con la versión anterior y además incluye un gran numero de mejoras y aspectos perfeccionados que hacen que disfrutes aun más de los juegos que desarrolles.
HAMMER Technologies
Alfonso Gómez 42, nave 112
28037 Madrid. España
Tel: (91) 3.04.06.22
Fax: (91) 3.04.17.97
Distribución en Argentina • Take Off Multimedia • Pueyrredon 495
Tel / Fax: (1704) 656 8506 • E-Mail: net2land@net2land.com

## Pueblos fantasmas

Por desgracia, también en el ambiente Div existen páginas que no se actualizan desde la época en la que el Fary era un sex symbol. Si las incluimos en nuestra revista es con la intención de hacer un llamamiento a sus webmasters.

DiVisor: Software, gráficos, freeware, Links, noticias, curso y E-Mail. Todo esto deberíamos encontrar en la página de Roberto Zamorano. Por desgracia, estamos ante una web que no se actualiza desde hace mucho. Y es una lástima, porque prometía. Así que, desde estas líneas lo dicho, un llamamiento al webmaster: ¡vuelve a ella!

DivGames Bilbao de Patxi Sánchez. Necesitan urgentemente grafistas, principalmente, y programadores para crear un grupo de desarrollo bajo el entorno de Div Games Studio. Lo necesitaban, porque el hecho es que nos encontramos ante otra página que lleva la tira de tiempo sin actualizarse. Que no decaigan los ánimos, que esta buena idea bien los merece.

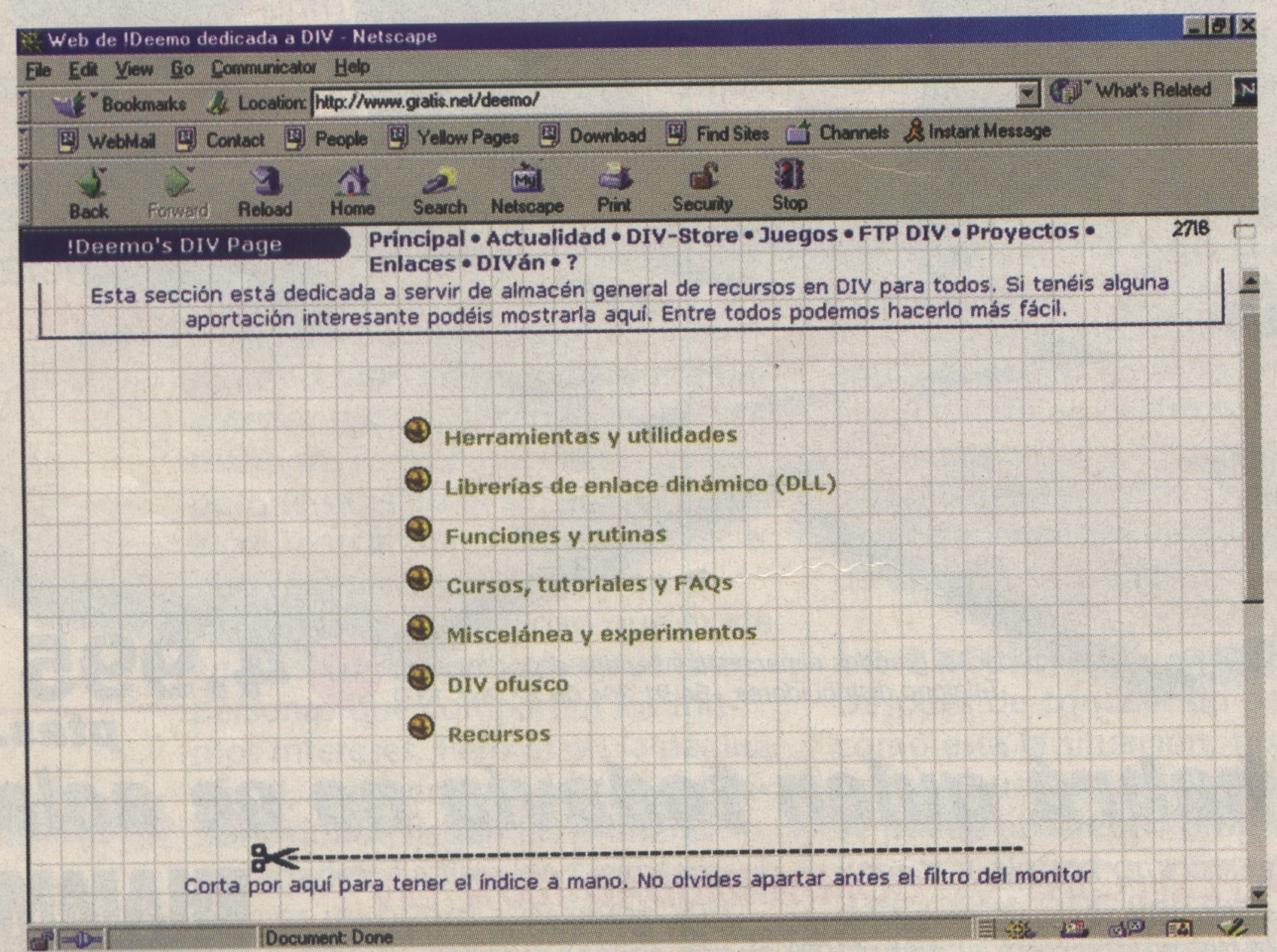
**Con lo bien que estaba planteada.**



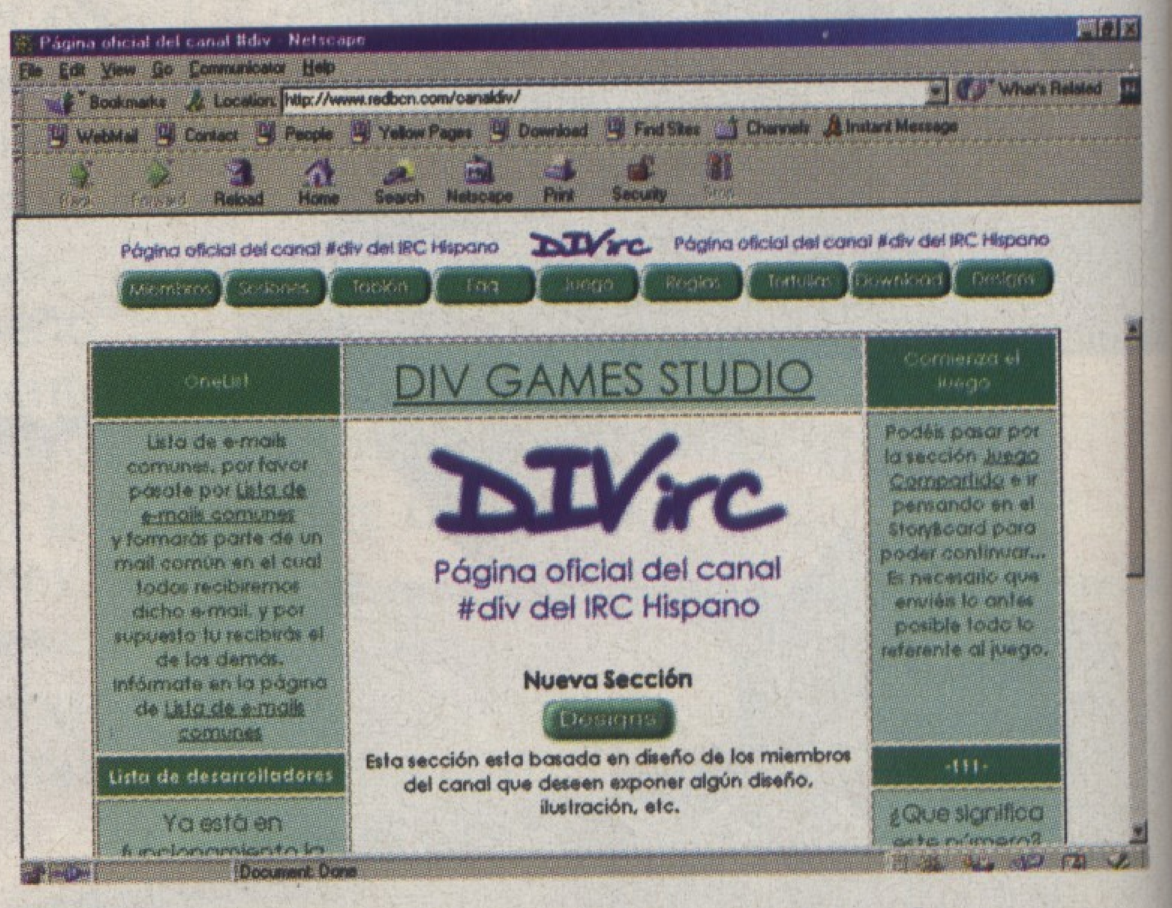
***Screenshot* de un juego hecho con *Creplat*.**

Y este mes (y van tres) os seguimos recomendando la página (Deemo=s DiV Page. Sencillamente, lo tiene todo: Actualidad, Div- Store (herramientas, recursos, utilidades, Juegos, Proyectos, etc.). Por si fuera poco, está bien diseñada. Imprescindible.

En la página oficial del canal #Div del IRC, DiV irc, vamos a encontrar utilidades y propuestas que interesarán sobretodo a quienes se hayan volcado en la programación en Div y busquen compañeros para involucrarse en un proyecto de altura. Y cuando decimos de altura no nos estamos refiriendo a un simulador de vuelo (a pesar de que nuestro humor es tan malo como para eso y más) sino a un juego de estrategia (el género que más votos ha conseguido de los miembros) aún muy en el aire, pero con alguna que otra propuesta interesante, como la de hacer un simulador de una ciudad utópica. Ilusiones (qué palabra más cursi), ganas de trabajar y compañerismo es lo que se respira por esta web que ya cuenta con más de 100 miembros. Además se organizan sesiones en las que se discute temas tan interesantes como la IA o los *scroolls* de los videojuegos, aparte de charlas de carácter didáctico. También llama la atención la sección *Designs,* donde encontraréis diseños de alguno de los miembros de esta página, o los numerosos y bien ordenados enlaces. En fin, una cita ineludible para la creatividad y el trabajo en grupo.

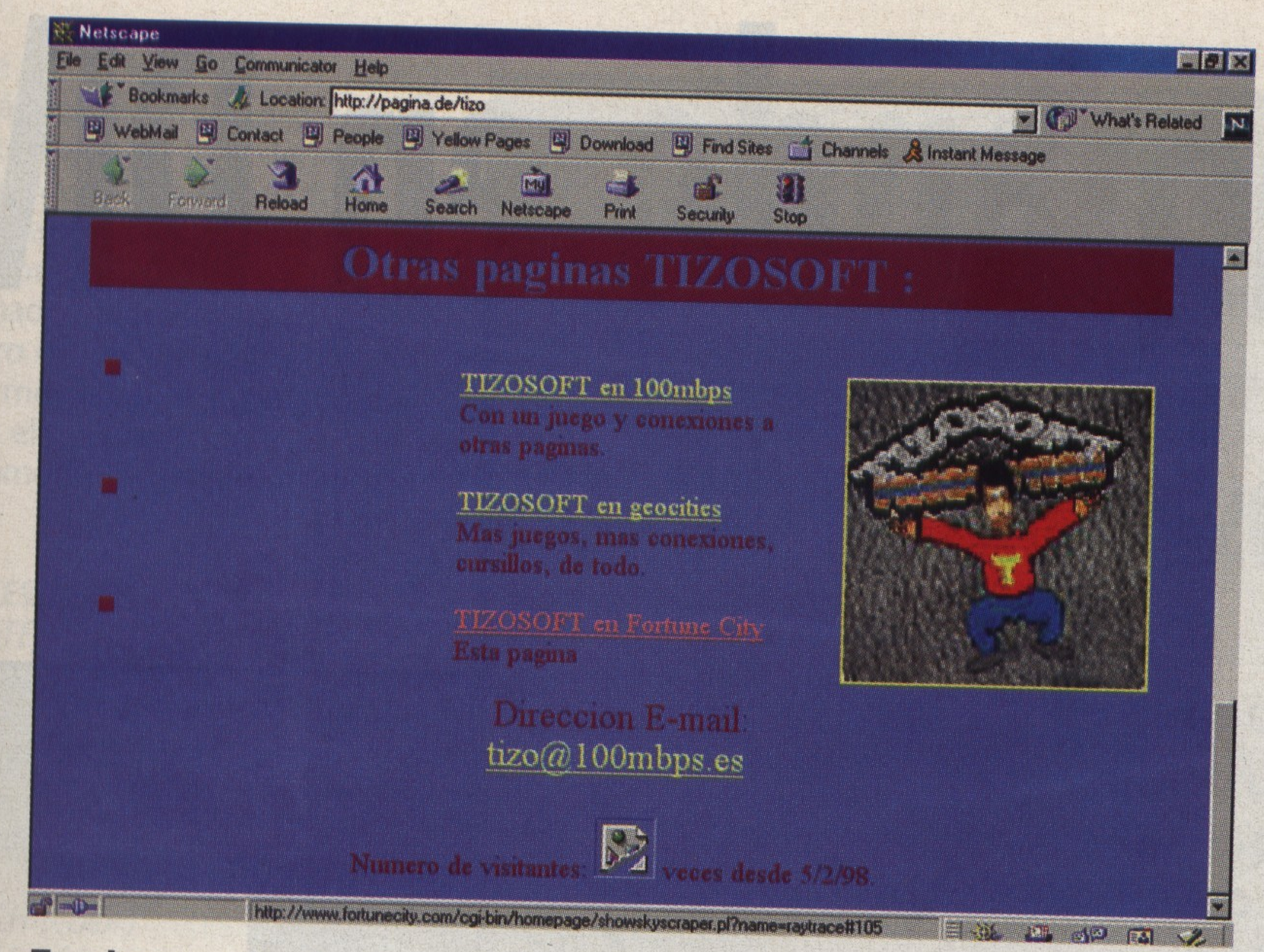



**Una buena biblioteca de recursos para Div.**

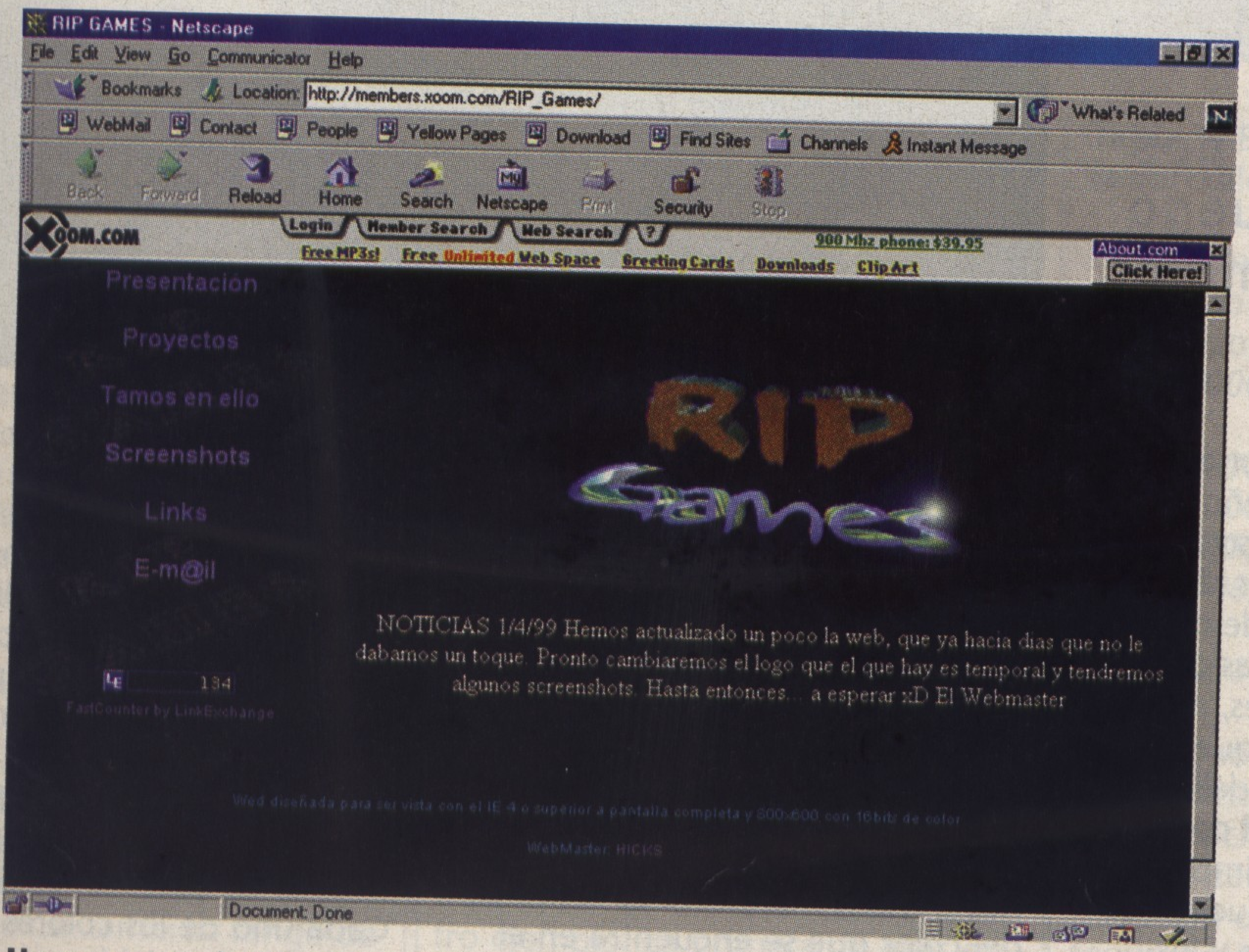



**Página principal de DiVirc, una web muy aconsejable.**

La página de Ximo es otra de las habituales de esta revista. Se trata del trabajo de un DiVero de pura cepa que tiene casi todo lo que un usuario de Div busca: juegos, noticias y *Links*. De estética, aceptable;




**Tres lugares donde encontrar a Tizo en la Red.**




**.Un proyecto interesante para una jovencísima desarrolladora.**

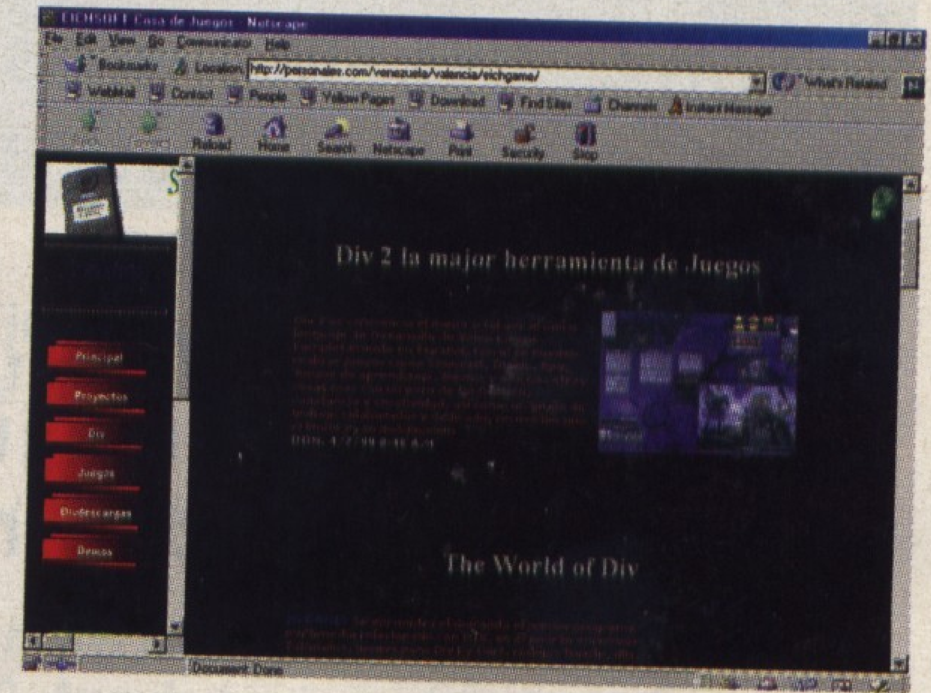
de contenido, más que un notable (incluso uno puede descargarse niveles del juego *Shodown)*

Y si hablamos de la página de Ximo, no vamos a dejar en el tintero la de Tizo, otro peso pasado en el mundo de Div Games Studio. Juegos para bajarse, recursos, interesantes programas y aplicaciones, propuestas, noticias y un curso de Div. Desde lo más básico hasta ese fichero que, por curioso, nunca habríamos imaginado que existe. Una advertencia: Tizo en la Red es uno y trino, ya que le podréis encontrar en Fortunecity (la dirección que aquí os señalamos), Geocities y 100 mbps. Merece la pena, por mucho que suban las tarifas telefónicas, pasarse por las tres.

## Grupos de desarrollo

¿Buscas una orientación acerca de todo lo que puede dar de sí Div? ¿Necesitas un grupo al que adherirte para poner en marcha o participar en un proyecto? Entonces, pásate por estos dos rincones:

Rip Games: Kabuto, Jordi y Hicks se han unido para crear un juego de estrategia (prometedor, con muchas ideas nuevas) bajo el entorno Div. En su página podremos encontrar noticias acerca de la evolución de este proyecto, así como capturas y otros documentos. Aunque, de momento, nada de nada. Esperemos que durante este verano encuentren el tiempo suficiente para dar un buen empujón a este título que, de momento, parece llamarse *Ruler of the world.*




**Parece que a éstos realmente les ha gustado Div.**

IG Game: proyectos, como no podían faltar en un grupo de desarrollo: una sección de Div, imprescindible para aparecer en estas páginas; Divdescargas, cuatro juegos y un pre-compilador con sus correspondientes *screenshots.* Un diseño más que aceptable termina por hacer de este rincón virtual un lugar más que aconsejable para visitar.

Ignacio Pulido

### Las direcciones de este mes

http://www.divgames.com: Página oficial de Div Games Studio
http://www.areaint.com: Área Interactiva.
http://www.babysoft.es/ddi: DDI
http://www.gratis.net/deemo/: (Deemo=s DiV Page.
http://www.redbcn.com/canaldiv: DiV irc
http://www.geocities.com/SiliconValley/Campus/6077/:Página de Ximo
http://pagina.de/tizo: Página de Tizo
http://members.xoom.com/RIP_Games/: Rip Games
http://personales.com/venezuela/valencia/eichgame/: IG Games
http://www.arrakis.es/~robzam: Divisor
http://web.jet.es/patxisanchez/: Div Games Bilbao

# Programando en DIV

## Las carreras y los mapas de durezas

**La técnica de los mapas de durezas es sin duda una de las más importantes en el mundo de la programación de videojuegos. Gracias a esta técnica, junto con el manejo de fuentes numéricas de texto, nos permitirá crear un juego de carreras mucho mejor que el anterior.**



En el número anterior desarrollamos un pequeño juego, con un simplísimo *scroll* que detectaba colisiones. En éste vamos a aprender el uso de mapas de durezas para crear zonas calientes o de colisión irregulares, mucho más potentes y versátiles que las colisiones, para así poder delimitar el terreno circulable por el coche de forma más real que una simple recta. Además, nos introduciremos en el mundo de los textos y los marcadores numéricos. Para ello realizaremos un pequeño juego en el que manejaremos un coche por un circuito cerrado. Pero antes de todo, pasemos a conocer qué son y cómo se usan los mapas de durezas.

### Mapas de durezas

Un mapa de durezas no es más que una copia de una imagen en la que coloreamos determinadas zonas con un mismo color para indicar que en ellas se realizará una acción especial. Tomando el ejemplo del número anterior, podríamos distinguir dos tipos de zonas distintas: la carretera y el césped. Por tanto, hacemos una copia de dicho mapa y coloreamos con un determinado color el césped y con otro la carretera. Sin embargo, este mapa nunca será visible sino que se utilizará en la trastienda para saber cuál es la acción que debemos realizar cuando el coche esté sobre una determinada zona. Pues bien, esto es un mapa de durezas; pero ¿cómo se usa? Pues es tan sencillo como pedir el color del punto deseado y según su valor actuar en consecuencia. ¿Y cómo podemos saber el color de esa zona? Para ello existe una función gráfica que es *map_get_pixel()*, que nos devuelve como valor el índice del color que se encuentra en el punto que indiquemos del mapa de durezas. Tomando este valor, podemos saber si el coche está o no sobre el césped y, por ejemplo, destruirlo en dicho caso. Posteriormente trataremos algo sobre los índices de color para poder comprender a fondo este tema.

La sintaxis de la función *map_get_pixel()* es la que sigue:

*Map_get_pixel(<fichero>,<gráfico>,<x>,<y>);*

Donde fichero es el índice asignado al fichero donde se encuentra el mapa de durezas (ya vimos su significado cuando hablamos sobre los fondos de pantalla); gráfico es el índice que le hemos asignado al gráfico del mapa de durezas dentro del fichero; x e y son las coordenadas del punto cuyo color queremos que nos devuelva la función.

Por tanto, ya tenemos el índice del color y estamos en disposición de actuar en consecuencia; pero antes vamos a tratar un poco el tema de las paletas de colores.

### Paletas de colores

DIV Games Studio dispone de un sistema de paletas de 256 colores. Cada uno de los colores está definido por 3 variables: rojo, verde y azul (RGB) que varían entre los valores 0 y 63. Con esto podemos definir más de 260.000 colores diferentes, pero siempre con el tope máximo de 256 colores simultáneos mostrados en pantalla. A cada uno de los colores de esta lista se le asigna un índice de 0 a 255 que lo determina. De esta forma, cada color tendrá cuatro propiedades: índice, rojo, verde y azul. Podemos ver estos valores pulsando sobre la opción Mostrar paleta del menú de paletas. Se nos abrirá una ventana donde tenemos los distintos colores situados en una ventana. Si pulsamos sobre ellos, a la derecha nos aparecerán los índices de color, así como los citados valores de color RGB.

Retomando el caso anterior de los mapas de dureza, el valor que

devuelve la función *map_get_pixel()* no es más que el índice propio del color situado en las coordenadas que le hemos indicado a la función. Por tanto, cuando rellenemos en el editor gráfico un área con un determinado color debemos anotar el número del índice para utilizarlo posteriormente en nuestro programa. Para ello, el método más sencillo y cómodo es usar la sentencia SWITCH:

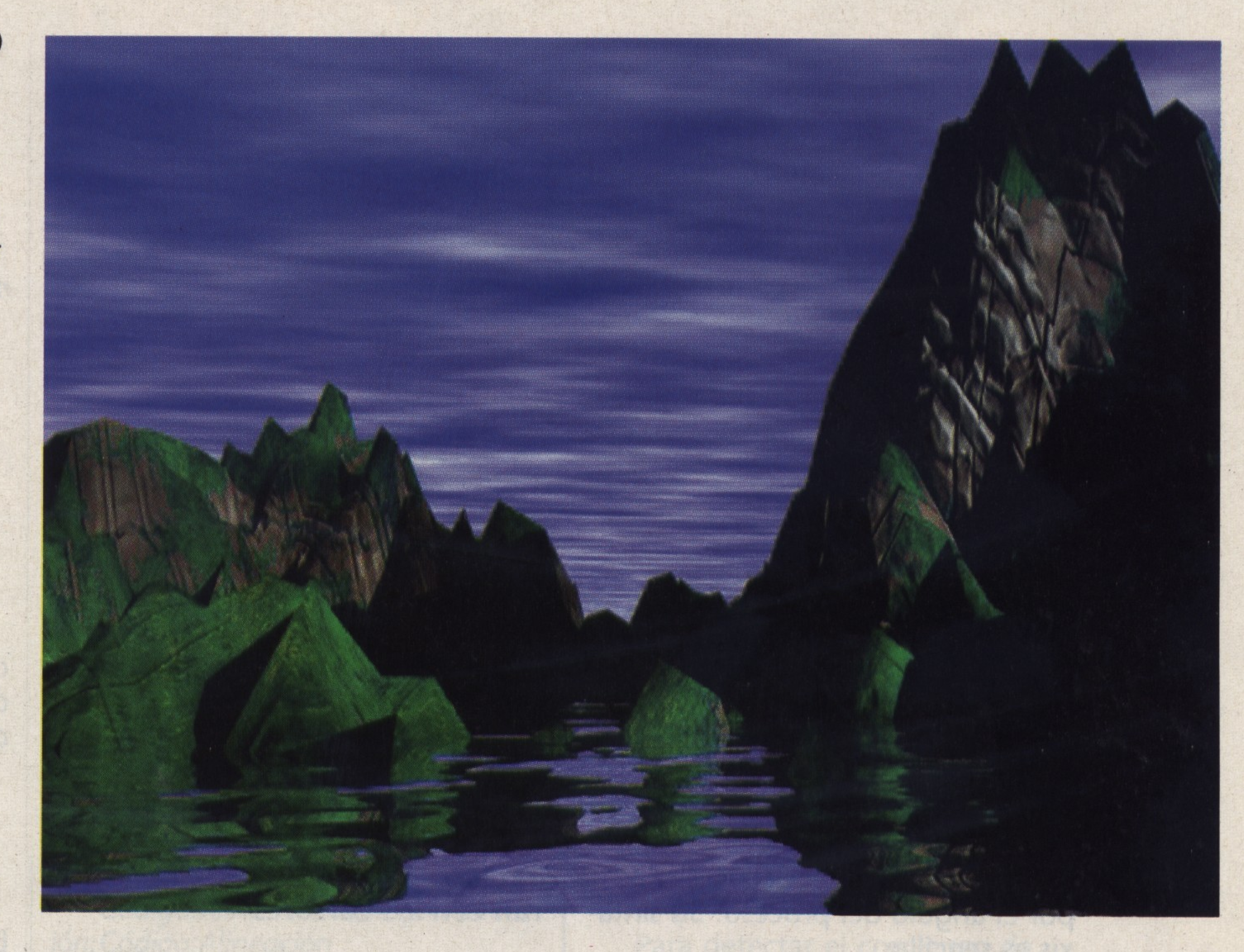
```
PROCESS proceso()
PRIVATE
Color;
BEGIN
//...
color = map_get_pixel (0,2,x,y);
// x e y indican la posición actual del proceso
SWITCH (color)
  CASE 200:
      // acción 1
  END
  CASE 201:
      // acción 2
  END
  DEFAULT:
      // acción por defecto
  END
 END
END
```

La sentencia SWITCH es una sentencia condicional que viene a reemplazar a una serie de *IF...ELSE* enlazada. La acción a realizar depende del valor de la variable que se encuentra entre paréntesis (color en este caso). Si el valor tiene una acción asociada, es decir, existe una sentencia CASE con dicho valor, se realizará esta acción y si no encuentra ninguna coincidencia, realizará la acción por defecto. Además de todo esto podemos indicar también rango de colores que ejecuten la misma acción. Para ello basta con sustituir el número de índice por un rango de la forma color1..color 2 indicando los límites inferior y superior. Por tanto la sintaxis completa de la sentencia SWITCH será:

```
SWITCH (<expresión numérica>)
  CASE <rango o valor>:
  // sentencias 1
  END
  ...
  DEFAULT:
  // sentencias por defecto
  END
END
```

## Mejorando los mapas de durezas

Es muy habitual, y de hecho recomendado, reducir el tamaño del mapa de durezas a la mitad. De esta forma reducimos la cantidad de memoria consumida para almacenar dicho mapa en un 75% (observad que donde anteriormente podía almacenarse tan solo un mapa de durezas, ahora caben 4). No perdemos ningún tipo de prestación y sólo debemos alterar la línea que comprobaba el valor del color con *map_get_pixel(a,b,x,y)* a la siguiente forma: *Map_get_pixel(a,b,x/2,y/2)*; Con ello ganamos velocidad y espacio sin apenas cambios.



## Ángulos y trayectorias

Para poder analizar el ejemplo de este número, vamos a analizar un par de cuestiones que serán de trascendental importancia en nuestro ejemplo. La primera de ellas es la variable local predefinida *angle*. Recordemos que una variable local es aquella que existe de forma independiente en cada uno de los procesos, y que al ser predefinida no debe ser declarada. Esta variable indica el grado de giro que tiene el mapa del proceso al mostrarse en la pantalla. Por defecto tiene un valor de 0; el valor está expresado en milésimas de grado, con lo que un ángulo de 1 grado estará expresado como 1000. Al ser una variable, para cambiar el valor de giro simplemente debemos tratarla como tal:

```
Angle=90000;
```

El valor 0 indica un ángulo hacia la derecha, y los subsiguientes positivos se encuentran en dirección contraria a la de las agujas del reloj (anti-horaria).

**Tabla 1. Ejemplo de valores para la variable *angle*.**

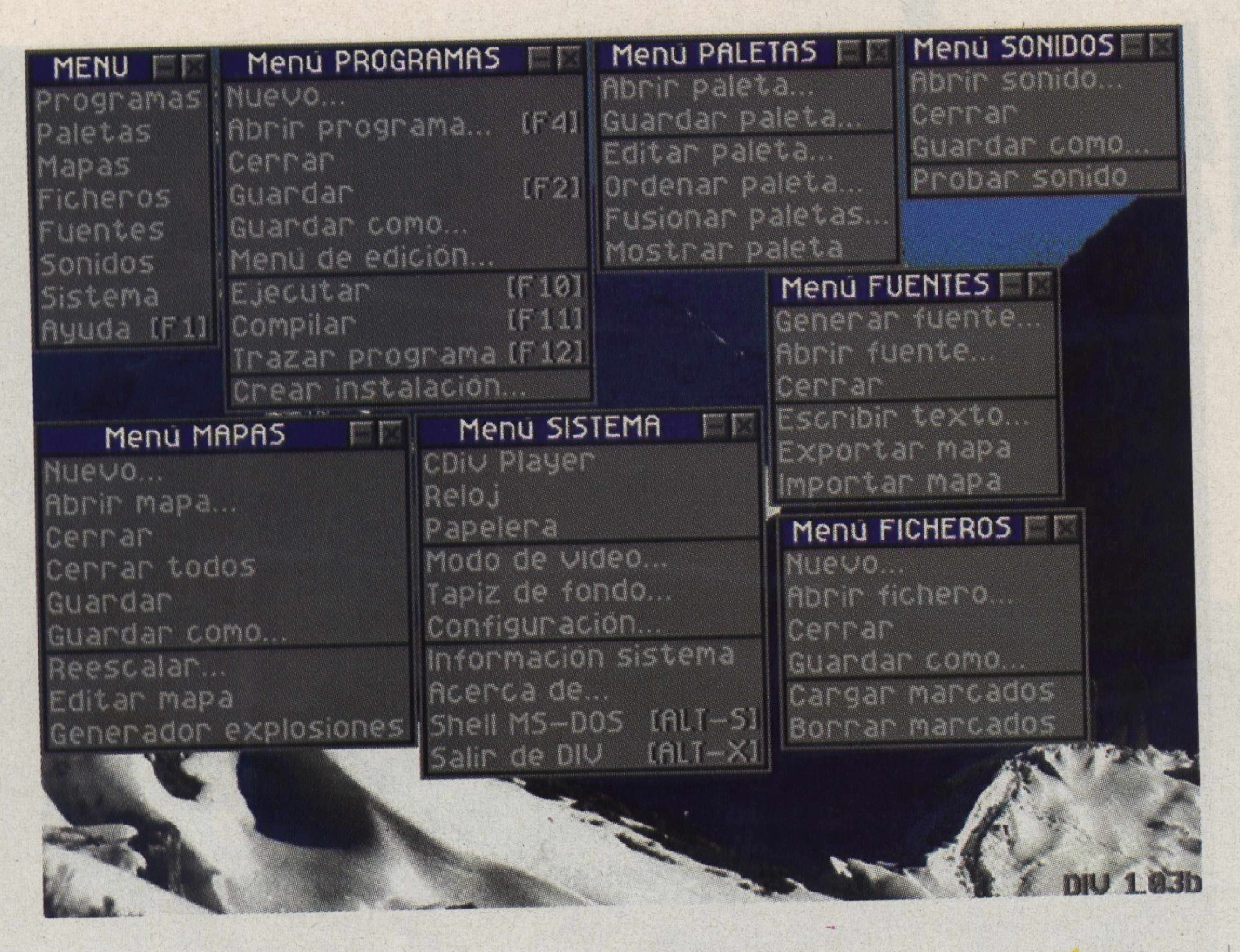
*Ángulo (valor de angle)*
*Significado*
*Ángulo hacia la derecha*
*+45000 Ángulo hacia la derecha / arriba*
*+90000 Ángulo hacia arriba*
*+180000 Ángulo hacia la izquierda*
*+270000 Ángulo hacia abajo*
*-45000 Ángulo hacia izquierda / abajo*
*-90000 Ángulo hacia abajo*

Por tanto si queremos realizar suavemente giros de 360 grados con el coche del ejemplo, basta con realizar un incremento o disminución de la variable *angle* según deseemos:

```
PROCESS coche()
BEGIN
Angle=0;
//...
LOOP
  Angle+=15000;
  FRAME;
;END
END
```

Recordemos que el signo += equivale a la sentencia *angle=angle+15000*. Análogamente podemos realizar lo mismo para el resto de operaciones (resta, multiplicación...).

Pero ¿qué hacemos para que el coche avance en la dirección que está señalando la variable *angle*? Para ello disponemos de la función *advance()* que indica al programa cuantos puntos queremos avanzar




en el sentido y dirección indicado por el ángulo del proceso. Su sintaxis es sencilla:

*Advance(<distancia>);*

Donde distancia es el valor del número de puntos que queremos avanzar. La función automáticamente varía el valor de las variables x e y modificando de forma instantánea la posición del proceso alterado.

Además de esta función, existen otras dos que conjuntas equivalen a la acción realizada por *advance()*. Se trata de *get_distx()* y *get_disty()*, que devuelven el número de puntos que se avanzarían en la dirección x e y respectivamente si se moviera un número determinado de puntos en la dirección del ángulo indicado. Su sintáxis es:

*Get_distx(<ángulo>,<distancia>);*
*Get_disty(<ángulo>,<distancia>);*

Donde ángulo indica el ángulo de avance y distancia el número de puntos a avanzar. Conjuntando ambas podemos realizar la misma acción que *advance()* mediante:

*x+=get_distx(angle,<distancia>);*
*y+=get_disty(angle,<distancia>);*

## Nuestro programa de ejemplo

Con estas dos funciones y los mapas de durezas, podemos realizar gran parte del programa ejemplo donde un coche recorre un circuito cerrado. El control se realizará mediante las teclas de cursor. Podemos ver qué acción realiza cada uno de los controles en la tabla 2.

**Tabla 2. Acciones que se realizarán con los cursores**

*Tecla (cursores) Acción*
*_up Incrementa en una unidad la velocidad*
*_down Disminuye en una unidad la velocidad*
*_right Reduce el valor de angle en 15000 milésimas de grado*
*_left Aumenta el valor de angle en 15000 milésimas de grado*

Para llevar el control de velocidad del coche utilizaremos una variable global (que bien podría ser privada del proceso coche) que indicará el número de puntos a avanzar en cada *frame*. Este valor será el que utilicemos en la función *advance()* para hacer mover el coche. Este valor no se verá alterado mientras se encuentre en carretera. En el momento en el que el coche pise los bordes rojo y blanco o se meta en el césped, no podrá avanzar más de 1 y 2 pixels por *frame* respectivamente. Para comprobar dónde pisa, recurriremos a la función *map_get_pixel()*. Analicemos el proceso coche del programa ejemplo:

```
PROCESS coche()
BEGIN
graph=11;
angle=90000;

x=540;
y=150;
LOOP
  IF (key(_up))
  velocidad++;
  END
  IF (key(_down))
  velocidad--;
  END
  IF (key(_left))
  angle+=15000;
  END
  IF (key(_right))
  angle-=15000;
  END
  Color =
map_get_pixel(0,2,x/2,y/2)
  SWITCH (color)
      CASE 230:
    velocidad=1;
    END
      CASE 245:
    velocidad=2;
      END
    END
  advance(velocidad);
  // aquí colocamos las sentencias para restringir el movimiento del coche a la pantalla sin que se salga de ella.
  FRAME;
 END
END
```

Los colores 230 y 245 indican el borde de la pista y el césped respectivamente. Como vemos, el valor de velocidad permanece inalterable a menos que nos salgamos de pista, donde existe una velocidad fija.

Por último vamos a añadir un marcador numérico en la pantalla que nos indique cuánto tiempo tardamos en dar una vuelta al circuito. Para ello, en primer lugar conozcamos cómo podemos crear un tipo de letra nuevo en función de los estilos predefinidos de DIV.

## Fuentes de texto

DIV dispone de toda una serie de funciones y opciones que nos permiten obtener unos resultados muy profesionales de forma sencilla. Además de disponer de una amplia librería de fuentes de texto ya hechas, podemos crear las nuestras propias a partir de unos estilos ya definidos. En nuestro ejemplo se puede utilizar cualquier tipo de letra que queramos; no obstante, aprenderemos cómo crear nuestro propio tipo de letra.

## Creando nuevas fuentes

Para crear una nueva fuente lo primero que debemos hacer es seleccionar la opción generar fuente... del menú de fuentes. Se nos abrirá un menú donde, en primer lugar, seleccionaremos el directorio y el nombre con el que grabaremos nuestro nuevo tipo de letra, pulsando sobre los puntos suspensivos situados junto a la opción nuevo fuente. Una vez hecho esto, buscaremos el aspecto que deseemos para nuestra fuente de entre los existentes en el directorio IFS. Durante la selección podremos

previsualizar el estilo, de forma que podamos orientarnos fácilmente.

Una vez seleccionados los estilos, el paso siguiente será determinar el tamaño de nuestra fuente. Para nuestro ejemplo usaremos 32x32, aunque podríamos utilizar cualquiera. Debemos remarcar que para tamaños menores de 16x16 no podemos especificar otros que no estén en la lista de la esquina superior derecha.

Ahora seleccionaremos el color de nuestra fuente, el reborde y la sombra. Para activar el reborde debemos indicar el número de puntos que tendrá y la dirección en la que aparecerá. Si queremos que aparezca en todas las direcciones debemos seleccionar el aspa. De forma parecida debemos indicar la sombra que utilizaremos. El primer número indica el desplazamiento horizontal de la sombra y el segundo el vertical.

Si queremos ver el trabajo realizado hasta el momento, debemos indicar en el recuadro de prueba los caracteres que queremos que se nos muestren. Acto seguido pulsaremos el botón de prueba y tendremos un resultado previo. Realiza los retoques pertinentes y una vez que la fuente esté a su gusto, pulsa en el botón generar y se creará automáticamente la fuente en el fichero indicado.

## Creando un marcador de tiempo en nuestro juego

Para colocar un marcador utilizaremos, obviamente, la fuente anteriormente creada. Por tanto en primer lugar debemos cargar en memoria la fuente de texto mediante la función *load_fnt()*. Ésta devolverá un código identificador de fuente que debemos recoger en una variable, ya que este valor se utilizará más tarde. Por tanto la sentencia de carga sería:

*Fuente=load_fnt(Anuevafnt.fnt@)*
*;*

Una vez cargada, recurriremos a la función write_int para mostrar el valor de una variable en la pantalla. Su sintaxis es la que sigue:

*Write_int(<fuente>,<x>,<y>,<ce ntrado>,<offset variable>);*

Donde fuente es el código que recogimos anteriormente en la carga de la fuente; x e y son las coordenadas donde queremos colocar la fuente numérica; centrado es uno de los valores indicados en la tabla 3 y offset variable indica la dirección de la variable que tiene el valor a mostrar. Para indicar la dirección de una variable, debemos anteponer al nombre la palabra clave OFFSET. ¿Para qué queremos saber la dirección de la variable y no su valor? Pues porque de esta forma el texto mostrado en la pantalla cambiará cuando el valor de la variable cambie; de este modo no debemos preocuparnos más de la presencia del marcador en la pantalla. Así un ejemplo que mostrase la velocidad del coche sería:

*Write_int (fuente,200,200,4,OFFSET velocidad);*

El valor mostrado variará de forma continua sin que debamos preocuparnos más de él.

**Tabla 3. Posibles valores que puede tomar el código de centrado de la función write_int().**

*Código:Alineación:Código:Alineac ión:Código:Alineación*
*0:Arriba izquierda:3:Izquierda:6:Abajo izquierda*
*1:Arriba:4:Centro:7:Abajo*
*2:Arriba derecha:5:Derecha:8:Abajo derecha*

Como dijimos al principio, queremos incorporar un contador de centésimas de segundo en nuestro juego. Para ello debemos tener en primer lugar un contador de tiempo, para lo que existe lo que se denomina una tabla global predefinida. No vamos a entrar ahora en detalles acerca de este tipo de datos, ya que no lo necesitaremos usar de forma inmediata. De forma rápida, una tabla viene a ser una lista de datos bajo un mismo nombre, de forma que a cada dato se puede acceder por un subíndice. Un ejemplo sería un vector con las coordenadas x, y, z asignándosele a cada una de ellas los valores 0, 1 y 2 respectivamente. Esta tabla global se llama *timer.* Tiene 10 valores, pero para acceder al primero de ellos basta con hacerlo simplemente usando la variable *timer* sin indicar ningún tipo de subíndices. El valor de *timer* indica el tiempo transcurrido en centésimas de segundo desde el comienzo del programa. Por tanto, para crear un marcador con este valor tan solo debemos enviar a la función *write_int* el valor del *offset* o dirección de la variable global *timer:*

*write_int(fuente,250,125,0, OFFSET timer);*



Con lo que ya tenemos el contador de tiempo. Ahora sólo nos falta poner el reloj a cero cuando se complete una vuelta. Para realizar esta acción debemos asignar el valor 0 a *timer* como si fuera una variable más ¿sencillo verdad?

*Timer=0;*

Para detectar el comienzo de una nueva vuelta, hemos colocado una raya ancha de color blanco (índice 15) de forma que cuando el proceso coche usando el mapa de durezas detecte que el coche pasa sobre ella, el reloj se pondrá a cero.

## Últimos retoques

El ejemplo de este mes a pesar de ser corto tiene un contenido denso y plantea una serie de técnicas de desarrollo básicas en cualquier videojuego. Está claro que podríamos mejorar el juego de manera significativa creando efectos de derrapajes, realizando otras acciones cuando el coche se sale, añadiendo un contador de tiempo de vuelta récord... Éstas sólo son algunas de las muchas cosas que con los conocimientos adquiridos hasta ahora podemos realizar. No obstante eso lo dejamos a cuenta de los lectores quienes, por cierto, pueden enviar sus sugerencias, mejoras, preguntas, críticas y demás a la dirección trinidad@arrakis.es

Para quien quiera desarrollar desde cero el programa ejemplo, indicar que el mapa del circuito se encuentra en el archivo coches2.map del directorio \coches. Ahí se encuentran todos los gráficos necesarios. Y si no, ¿alguien se atreve a crear un nuevo circuito con su correspondiente mapa de durezas? Ánimo y buena programación.

Pablo Trinidad

# Archivos PCX (I)

## Manipulando imágenes

**Vamos a ver cómo podemos realizar funciones para visualizar archivos PCX en pantalla, así como todo lo que necesitaremos para realizar diversos efectos en la pantalla de nuestro juego.**




Para crear efectos visuales necesitamos de diversas funciones y punteros de direcciones que nos permitan acceder principalmente a la pantalla, al buffer posterior, al fondo de pantalla y a las paletas. En primer lugar analicemos cuál es el proceso que se suele seguir en cualquier secuencia de imágenes para mostrarse en la pantalla de forma fluida.

### Secuencias de animación.

En toda animación se sigue un cierto proceso para que sea lo suficientemente fluida como para que no se perciba parpadeo. El parpadeo se produce por lo que se llama velocidad de refresco. La pantalla se dibuja de forma horizontal, mediante un recorrido de un haz que obviamente tarda un tiempo, corto eso sí, en completarla. Si cambiamos el contenido de la memoria de vídeo mientras se está actualizando, cuando el haz complete su recorrido habrá un trozo de la anterior pantalla y otro de la nueva. En el siguiente haz se recuperará el otro trozo nuevo no mostrado, pero se producirá un leve parpadeo. Si esto lo aplicamos a una animación en la que se muestren por ejemplo 60 imágenes por segundo, es normal que se produzca este fenómeno de parpadeo de forma excesivamente frecuente, con lo que la animación se ve de forma escalonada. Yendo más allá, si dibujamos directamente muchos procesos en la pantalla, este parpadeo se hará insoportable, además de ralentizar sobremanera la animación. Para ello se recurre a lo que se le denomina *flipping*. Esta técnica consiste en realizar todas las variaciones y actualizaciones necesarias de la pantalla en una pantalla virtual, es decir, en una zona de memoria que simula a la pantalla principal y del mismo tamaño en pixels, y que obviamente no se ve. A esta pantalla virtual se le denomina *buffer* posterior *(backbuffer)*. Una vez realizados todos los cambios del nuevo frame o secuencia de animación a mostrar en el *buffer* posterior, podemos mostrarla ya directamente en la pantalla. De esta forma reducimos el parpadeo. Pero sigue existiendo. Para eliminarlo, basta con sincronizarlo con la frecuencia de refresco de la pantalla (refresh rate) de forma que cuando ésta empiece a recorrer la pantalla, el nuevo frame ya esté colocado en la memoria de vídeo.

### Acceso al proceso de animación

Por tanto, para interceder en todo este proceso descrito anteriormente, necesitamos 3 datos o direcciones de memoria fundamentales:

- Memoria de vídeo: puntero a la dirección de la memoria de vídeo donde se encuentra la imagen que se muestra actualmente en la pantalla. DIV no facilita dicho puntero de forma directa por posibles manipulaciones incorrectas que puedan provocar fallos en el sistema. Aún así, se pueden implementar funciones para un solo modo de pantalla utilizando directamente esta dirección. Por ejemplo para el modo MCGA 320x200 de 256 colores, y para otros muchos modos, es 0xA000 (en hexadecimal).
- *Backbuffer:* puntero al *buffer* posterior o pantalla virtual donde realizaremos las manipulaciones pertinentes para mostrar posteriormente la imagen por pantalla. Podemos acceder a esta dirección mediante el puntero *char *background.*
- Imagen de los procesos: su acceso es mucho más complejo. Aún así, podemos acceder al puntero de fondo de pantalla y mani-

# DIV developer

NÚMERO 3

## Sumario






## Curso de programación y concurso

Continuamos con nuestros cursos realizados con la única finalidad de enseñar a programar a nuestros lectores. Hemos podido comprobar que esta sección es una de las preferidas de nuestros lectores. Muchos de ellos se han iniciado en el mundo de los videojuegos gracias a DIV y ahora quieren ampliar sus horizontes; otros ya tenían conocimientos previos y quieren saber más, y, como no, también habrá los que, siendo profesionales, quieran echar un vistazo a los conceptos básicos. Al fin y al cabo nunca viene mal repasar nuestros conocimientos, sobre todo cuando hace mucho tiempo que fueron adquiridos y corren el riesgo de oxidarse en nuestro cerebro.

Como es lógico continuamos con los cursos iniciados en el número anterior de nuestra revista. Como ya sabéis, nuestro primer curso habla de la programación entendida en sentido amplio. Para ello continúa su repaso a la programación básica, es decir la programación basada en algoritmos.

El segundo curso también sigue la línea iniciada en Divmanía 2 introduciéndonos en la programación en C, un lenguaje de programación que ya es algo más que un clásico, aunque mantiene su vigor gracias a sus excepcionales características.

Por último, tenemos el curso de programación en Ensamblador donde se continúa dando cuenta de todo lo que tiene que ver con el lenguaje que utilizan los microprocesadores de nuestros ordenadores.

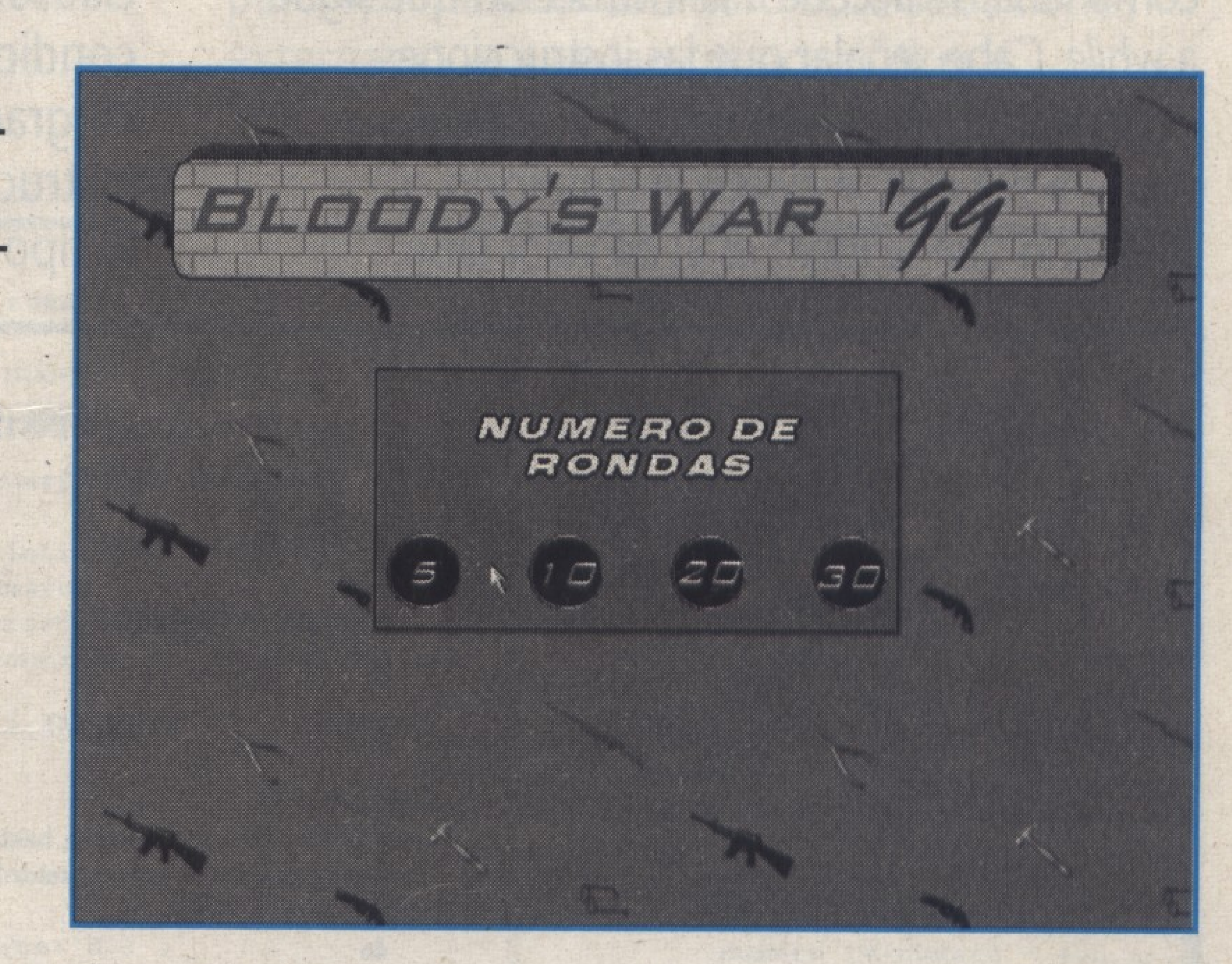
Y como no, os damos cuenta de los tres títulos que han resultado elegidos en nuestro concurso de programación con DIV. En nuestras páginas encontraréis cumplida información sobre ellos, con especial atención lógicamente a los aspectos de programación de los tres juegos.

Por supuesto, nuestro concurso sigue en pie con los mismos premios. Ya sabéis: 25.000 pesetas para el ganador y 20.000 para los otros dos ganadores. Esperamos con impaciencia recibir vuestros juegos. Ya sabéis dónde mandarlos, pero por si acaso hay algún despistado, os recordamos que podéis hacerlo por e-mail o correo normal:

Nuestra dirección:
Divmania@prensatecnica.com

Divmanía
C/ Alfonso Gómez, 42
Nave 1-1-2
28037, Madrid, España




## Destacamos

**En nuestro CD de portada incluimos las siguientes demos, que han resultado ganadoras en el concurso que realizamos entre los lectores:**

- **Disco Fighter, un programa de lucha muy interesante.**
- **PA-PÚ, un título altamente adictivo.**
- **Bloody's War'99, un juego no apto para impacientes.**

# Instrucciones

**Dentro de este número continuamos explicando las diferentes instrucciones de control de programas que son imprescindibles dentro de la programación básica. Los algoritmos ya no tienen secretos para nosotros.**

A continuación veremos otras instrucciones que hay que tener en cuenta a la hora de acercarse a la programación básica.

## INSTRUCCIONES ITERATIVAS

Permiten la ejecución de una secuencia de instrucciones de forma reiterada, dependiendo del valor de una condición. Es habitual que los lenguajes de programación incluyan sus tres tipos: *while, repeat* y *for,* aunque con sólo disponer de una de las anteriores resulta factible simular el comportamiento del resto.

**Las instrucciones iterativas permiten ejecutar secuencias de instrucciones de forma reiterada**

Una instrucción de tipo *while* comienza evaluando la condición que tiene asociada. En el supuesto de que sea considerada verdadera, pasa a activar el conjunto asociado de instrucciones. Al terminar por fin de ejecutarse se vuelve a analizar la condición y así sucesivamente. Cuando ésta queda determinada como falsa, se accede a la instrucción que sigue a *while.* Cabe señalar que las instrucciones pertenecientes a *while* podrían no ejecutarse si la primera vez que se evalúa la condición el resultado que se logra es el de valor falso.



```
FORMATO PARA LA INSTRUCCIÓN
WHILE

while condición
    do
      bloque_instrucciones

EJEMPLO

......
contador := 5 ;
{este bucle se ejecuta 5 veces, hasta que
 la variable contador tome el valor 10}
while contador < 10
    do
      contador := contador +1 ;
writeln (contador) ;
......
```

*Figura 5.*

En Pascal la anterior instrucción se denomina de igual forma. En la figura 5 es posible encontrar el diagrama de flujo asociado, el formato correspondiente en Pascal y un ejemplo sobre su utilización. Resulta normal encontrar en otros lenguajes de programación instrucciones iterativas cuyo nombre es *while,* pero que se corresponden con un tipo de instrucción diferente. Por eso, lo más conveniente en el caso de que se esté empezando a programar, es consultar el manual del usuario del correspondiente lenguaje para conocer la lógica de la instrucción.

Las de tipo *repeat* empiezan por ejecutar el conjunto de instrucciones que le acompaña, para a continuación evaluar su condición. Si ésta se determina falsa, se ejecutan de nuevo las instrucciones asociadas al bucle. Si por el contrario la condición se evalúa verdadera, se activa la siguiente instrucción. La lógica asociada a ella asegura que las instrucciones vinculadas al bucle se ejecutarán al menos una vez.

En Pascal, este tipo de instrucción se denomina de igual forma, quedando delimitadas las instrucciones asociadas por la claúsula *until* donde se encuentra la condición. En la figura 6 se muestra el diagrama de flujo asociado, el formato de la instrucción y un ejemplo que muestra su uso.

El tipo de instrucción *for* basa su ejecución en una variable que hace la función de contador. Cuando comienza a ejecutarse se inicializa dicha variable y justo entonces se comprueba la condición asociada al bucle. Si ésta se evalúa como verdadera, entonces se procede a ejecutar el conjunto de instrucciones asociado a *for,* para posteriormente incrementar o disminuir la variable (que hace las funciones de contador) en

```
FORMATO PARA LA INSTRUCCIÓN
CASE

Case variable of
 lista_valores_1 : bloque-1 ;
 lista_valores_2 : bloque-2 ;
       ....
 lista_valores_n : bloque-n ;
 [else bloque]
end ;

EJEMPLO

{la variable opción es de tipo char}

case opcion of
 'A', 'a' : writeln ('Selección de la opción A'
 'B', 'b' : writeln ('Selección de la opción B'
 'C', 'c' : writeln ('Selección de la opción C'
  else
    writeln('Opción errónea') ;
 end ;
```

*Figura 4.*

un valor especificado por una expresión q se habrá fijado antes de la ejecución del bucle. Esta descripción es general y no to los lenguajes permiten, por ejemplo, especificar el incremento de la variable o condición de parada para el bucle, caso d Pascal y otros lenguajes.

**Las instrucciones asociadas una instrucción iterativa no modifican los elementos que forman la condición**

En Pascal la instrucción *for* se inicia asignan un valor original a la variable contador (de ser de tipo entero u ordinal). Si dicho valo supera al valor límite, entonces se procede ejecutar el conjunto de instrucciones asociadas al bucle y automáticamente el b procede a incrementar o decrementar la variable contador en 1, procediendo a comprobar que la variable contador no ha superado el límite. Para indicar si se decrementará o incrementará la variable q controla la ejecución de esta instrucción, se utilizan respectivamente las palabras reservadas *downto* y *to.* No se permite modificar el límite durante la ejecución de esta instrucción, así como asignar ningún valor a la variable contador, aunque ésta puede formar parte de expresiones dentro las instrucciones del bucle. En la figura 7 se encuentra el formato que tiene esta instrucción en Pascal, al igual que un ejem para su utilización.

Es importante indicar que si las instruccion asociadas a una instrucción iterativa no modifican los elementos que forman parte la condición, se producirá un bucle infinito Esto quiere decir que el programa, una vez ha pasado a ejecutar las instrucciones asociadas a la instrucción iterativa, no pod

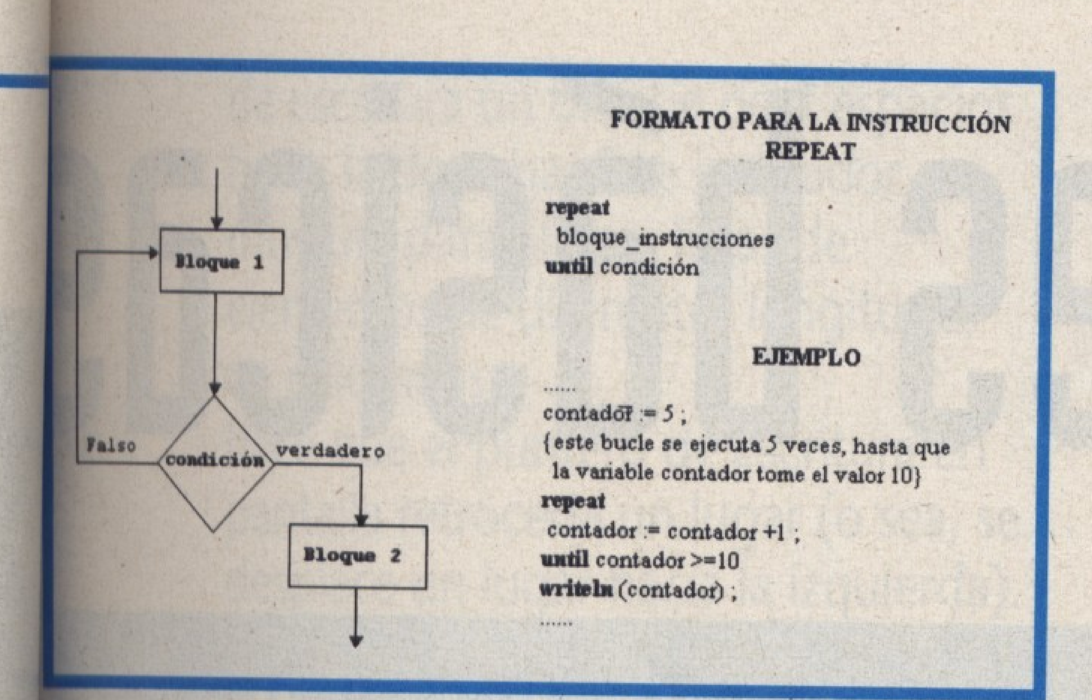



*Figura 6.*

salir del bucle. Por lo tanto, se debe tener cuidado a la hora de utilizar estas instrucciones.

## ÁMBITO DE LAS INSTRUCCIONES

Hasta ahora se han descrito las instrucciones básicas que todo lenguaje de programación procedural posee y que pueden denominarse en conjunto como "instrucciones de control de programa" (afectan al orden en que se ejecutarán las instrucciones). Todas estas instrucciones poseen la cualidad de poder tener asociada una o varias instrucciones. Al conjunto de aquellas que afectan las instrucciones de control se les denominará ámbito de la de control.

### Hay que tener en cuenta que toda instrucción pascal finaliza con punto y coma

En todo lenguaje existen una serie de caracteres especiales o palabras reservadas que permiten delimitar el ámbito de una instrucción de control, que en el caso de Pascal son *begin* y *end.* Entre ellas se debe situar el conjunto de instrucciones que se verán afectadas por la instrucción de control. A este conjunto se le denomina bloque de instrucciones. Si la instrucción es única, entonces no es necesario la utilización de estos delimitadores. Es importante verificar que para cada *begin* existe su correspondiente *end.* De hecho, un programa indica su ámbito en Pascal a través de las mismas palabras reservadas, pero teniendo en cuenta que la palabra *end* debe ir acompañada de un punto. También hay que indicar que toda instrucción Pascal finaliza con un punto y coma, por lo que a todo *end* debe acompañarle dicho carácter (salvo en el fin de programa y antes de una claúsula *else*).

```
FORMATO PARA LA INSTRUCCIÓN
FOR

for variable_control := valor_inicial [to|downto]
   valor_final
   do
    bloque_instrucciones

EJEMPLO
......
contador := 3 ;
{este bucle se ejecuta 5 veces, al finalizar
 la variable contador tomará el valor 8}
for i := 1 to 5
  do
   contador := contador + 1 ;
writeln (contador) ;
......
```

*Figura 7.*

Otros lenguajes utilizan en vez de palabras reservadas los caracteres { y }, para especificar un bloque (por ejemplo, C) y otros tienen para cada instrucción de control una palabra reservada que indica el ámbito de cada una de ellas (por ejemplo, Basic y Clipper). Hay que indicar de nuevo que siempre que se especifique un comienzo de bloque de instrucciones, también deberá especificarse el final del mismo. Para las personas que se están iniciando en las técnicas de programación, ésta suele ser una fuente de errores de compilación y ejecución bastante frecuente.

Para ayudar a la comprensión del programa y poder establecer qué bloques de instrucciones pertenecen a qué instrucciones, es útil la costumbre de indentar los programas. La indentación consiste sencillamente en establecer un margen común para aquellas instrucciones que pertenezcan a un mismo bloque. Dicho margen estará tanto más a la derecha cuanto mayor sea el anidamiento de las instrucciones.

Otra buena costumbre a la hora de utilizar las instrucciones de control de programa consiste en realizar comentarios a las condiciones asociadas a dichas instrucciones.

## OTRAS INSTRUCCIONES

Hasta ahora se han descrito las instrucciones comunes a casi todos los lenguajes de programación, pero puede ocurrir que no tengan la misma denominación, que no posean el mismo formato, aunque sí exista una correspondencia entre su lógica. De todas estas instrucciones falta describir la instrucción *goto.* Esta instrucción cayó en desuso debido a que, utilizada de forma inapropiada, tiende a hacer los programas ilegibles. De hecho, no hay situaciones de programación que necesiten de uso.

La instrucción *goto* transfiere la ejecución del programa a una sentencia marcada por la etiqueta que acompaña a la instrucción. La única situación en que parecería razonable utilizar esta instrucción es para salir de un bucle en el que se están llevando a cabo una serie de operaciones, donde la condición de salida es muy compleja y no se disponen de instrucciones extra para salir del bucle. De todas formas, esta situación suele estar provocada por un mal diseño del programa y seguramente puede ser subsanada revisando dicho diseño, lo que siempre es preferible a utilizar dicha instrucción. De todas formas, si se utiliza, es aconsejable incluir algún tipo de comentario que indique la función que realiza o la condición que provoca su ejecución.

### La instrucción goto tiende a hacer los programas ilegibles si se usan inapropiadamente

Pueden existir además una serie de instrucciones, que permiten modificar el flujo de la lógica en un programa, estas instrucciones son las siguientes: *break, continue, exit* y *halt.*

La instrucción *break* finaliza la ejecución de una instrucción iterativa, sin atender a la condición asociada a ésta. La instrucción *continue* prosigue con la ejecución de la siguiente iteración de un bucle, aunque éste no haya finalizado la ejecución de la totalidad de su bloque asociado. La instrucción *exit* sirve para salir de forma inmediata del procedimiento, función o programa en el que dicha instrucción se encuentre situado. Por último, la instrucción *halt* detiene de forma inmediata la ejecución del programa y regresa al sistema operativo.

Normalmente, estas instrucciones se utilizan para optimizar el código o ante situaciones externas que puedan provocar el mal funcionamiento del programa.

```
Program num_romanos ;
 var
  cociente, numero : integer ;
  contador : integer ;

begin {del cuerpo del programa}
 repeat
   writeln('Introduzca un número') ;
   readln(numero) ;
 until (numero>0)and(numero<4000) ;
{numero ya contiene un valor entre 1 y
3999}
 if numero >=1000
   then
     begin
      cociente := numero div 1000 ;
      for contador := 1 to cociente
         do
          write('M') ;
      numero := numero -1000*cociente ;
     end ;

if numero >= 900
   then
    begin
     write('CM') ;
     numero := numero - 900 ;
    end ;
   else
    if numero > =500
      then
       begin
        write('D') ;
        numero := numero - 500 ;
       end
      else
       if numero >= 400
         then
          begin
           write('CD') ;
           numero := numero - 400 ;
          end ;

          . . . . . . . . . . .
```

*Figura 8.*

# PROGRAMACION EN C

# Variables y funciones básicas

Seguimos indagando en los diferentes tipos de funciones y variables del lenguaje C, conceptos fundamentales para entender este conocido sistema

**La programación en C sigue siendo uno de los pilares básicos dentro de la creación de programas. En este nuevo capítulo de nuestro especial cursillo sobre el tema analizaremos algunas de las funciones más útiles para utilizar este lenguaje.**

## INICIALIZACION DE VARIABLES

Como ya se explicó en el anterior capítulo, las variables se definen escribiendo el tipo que queremos definir (uno de los expuestos anteriormente) seguido del nombre de la propia variable.

Hasta aquí es fácil de entender (como se puede comprobar en los ejemplos dados más arriba), pero además tenemos que las variables se pueden inicializar con un valor inicial para que posean de esta forma un valor cuando se inicie el programa.

**La función *printf()* sirve para mostrar en pantalla todo tipo de textos e información**

La asignación de valores es simple de realizar, sólo tenemos que poner tras el nombre de variable un signo igual y después el valor que le queramos dar. Ejemplos:

```
int variable1 = 200;
long variable2 = 35042304;
float variable3 = 2344.33;
```

## LA FUNCION PRINTF

La función *printf()* sirve para poder mostrar en pantalla toda clase de información, pudiendo hacer que escriba textos, números enteros y de coma flotante tanto en decimal, hexadecimal o incluso octal, etc.

La base de su funcionamiento es la de imprimir la cadena de texto que se indique dentro de los paréntesis, aunque dicha cadena, a su vez, puede contener muchos otros elementos. Pongamos un ejemplo sencillo:

```
int EDAD = 29;
printf("Mi edad es %d"; EDAD);
```

En el código dado de ejemplo realizaría la función de escribir en pantalla la cadena <mi edad es 29>.

Como se puede observar en la instrucción, los caracteres %d han sido sustituidos a la hora de escribir la cadena por 29. Ello es debido a que el caracter % se usa para indicar que lo que hay a continuación es uno o más caracteres que indican el tipo de dato que debe escribirse en ese lugar de la cadena de texto.

En el caso concreto del ejemplo dado, %d significa que se ha de escribir un entero decimal con signo. El dato, por supuesto, deberemos proporcionárselo también nosotros, y deberemos colocarlo como parámetro extra en la propia función dentro del paréntesis después de la propia cadena.

En el caso del ejemplo, el dato proporcionado ha sido la variable EDAD, que poseía el valor 29, ya que es el valor con el que se había inicializado en la línea anterior al definir EDAD.

Todo lo explicado se puede resumir en lo siguiente: se ha conseguido sacar el valor de la variable EDAD por pantalla.

## LOS INDICADORES DE TIPO

Además de %d, existen muchos otros tipos de indicativos posibles a colocar tras %. Son los siguientes:

- %c = escribir un caracter por pantalla.
- %s = escribir una cadena por pantalla.
- %d = escribir un decimal por pantalla.
- %i = escribe un número en formato decimal con signo.
- %f = escribe un número de punto flotante en formato decimal
- %e = escribe un número de punto flotante en notación exponencial.
- %g = escribe un número en punto flotante.
- %u = escribe un número decimal sin signo.
- %x = escribe un número en hexadecimal sin signo.
- %o = escribe un número en formato octal.
- l = prefijo que se usa junto con %d, %u, %z, %o y %i para indicar que lo que se quiere escribir son enteros largos y también con %f, %e y %g para indicar números flotantes de doble precisión.
- h = prefijo usado con %d, %u, %x, %o y %i para indicar que se trata de un entero corto.

**Existen muchos tipos de indicativos posibles que se colocan tras %**

Otra propiedad de *printf()* es que dentro de la cadena base se pueden incluir tantos elementos como se quieran, pudiendo indicar por ejemplo 3 ó 4 elementos %d, mezclar varios tipos o hacer la combinación que queramos o necesitemos en un momento determinado. Ejemplo:

```
short ANYO = 1940;
int EDAD = 29;
char GRUPO = "A";
printf("Hola, nací en el año %d, tengo %d años y mi grupo sanguíneo es el %c", ANYO, EDAD, GRUPO);
```

## CODIGOS DE ESCAPE

Además del caracter % también existe otro, considerado por *printf()* como un indicador especial. Este indicador es "\", la barra invertida, y se usa para indicar las llamadas secuencias de escape, que permiten por ejemplo que el puntero de escritura cambie de línea, que se produzca una tabulación en ese punto, etc. Las secuencias de escape disponibles son las siguientes:

- \n = se usa para hacer pasar al puntero de escritura en pantalla a la línea siguiente y para que se coloque el cursor en el margen izquierdo.
- \t = se usa para que se tabule el puntero

de escritura un bloque de 8 espacios hacia la derecha. Este indicador es muy útil para alinear listas de elementos de diferentes longitudes en columnas.

\b = hace que el puntero de escritura en pantalla retroceda un lugar (o sea, se desplace un lugar hacia la izquierda).

\r = realiza un retorno de carro.

\f = hace que el puntero de escritura vuelva al inicio de la primera línea de pantalla.

\' = hace escribir una comilla simple.

\" = hace que se escriba una comilla doble.

\\ = se escribe una barra invertida.

\xdd = se escribe en pantalla el caracter ASCII indicado en dd, que será un número hexadecimal.

\ddd = se escribe en pantalla el caracter ASCII indicado en dd, que estará indicado en octal.

## Se considera que la función scanf () es la más usada dentro del lenguaje C

Para los que estén un poco despistados respecto a lo de la tabla ASCII, recordar que en el anterior capítulo ya se explicó que se trataba de un conjunto de caracteres estandarizados que siempre poseen la misma equivalencia numérica.

### LA FUNCION SCANF

La función *scanf()* es considerada la función de entrada de datos más utilizada en el lenguaje C. A continuación se da un ejemplo de programa que usa *scanf():*

```
main( )
{
 float anyos, dias;
 printf(" Por favor escriba su edad en años");
 scanf("%f", &anyos);
 dias= anyos*365;
 printf("Usted ha vivido &f días.\n", días);
}
```

Excepto la línea *scanf,* el resto del programa tendría que ser ya comprensible más o menos para el lector.

Primero se inicia el programa con *main().*

Después se definen dos variables (*anyos* y días), a continuación se escribe el texto por pantalla <Por favor escriba su edad en años> y después llega *scanf(),* que sirve para que el usuario pueda escribir su edad, quedando ésta almacenada en la variable *anyos* una vez se pulse Enter.

El formato usado en *scanf()* es idéntico a *printf():* se escribe en la cadena que se incluye como parámetro el indicador del tipo de dato del que se quiere pedir un dato tipo *float* y como parámetro segundo se ha indicado la variable precedida de & en la que queremos que se guarde el valor entrado por el usuario.

Se ha usado la variable tipo *float* porque el usuario quizás quiera introducir su edad en años usando también decimales (por ejemplo 29.5), de esta forma no se perderá precisión.

Una vez entrado el dato, el programa coge los años introducidos (anyos) y los multiplica por los días que tiene un año (365).

Hecho el cálculo y almacenado en días, sólo resta escribir el resultado por pantalla, para lo cual nada más fácil que usar un *printf()* indicando el tipo de variable a escribir (float = %f).

El detalle de que se tenga que escribir un signo & delante de cada variable que indiquemos en *scanf()* es un tema relacionado con la memoria y de momento se va a saltar su explicación, aunque más adelante ya se hablará del tema.

### EL OPERADOR AMPERSAND (&)

La función *scanf()* como puede observarse, usa en el programa de ejemplo un nuevo símbolo no explicado hasta ahora, es el llamado *Ampersand* (&) y el cual precede al nombre de variable.

La explicación de su presencia es simple de entender. El carácter & se ha colocado porque la función *scanf()* no requiere como parámetros las propias variables que va a usar, si no las direcciones de memoria física que ocupan.

Dicho lo anterior, queda claro que la utilidad del signo *Ampersand,* es la de indicar al programa la dirección de memoria en la que está ubicado el elemento que se indica inmediatamente después, y así pues, si indicamos &var1, lo que hacemos es dar la dirección de memoria donde está almacenada la variable.

Ejemplo que visualiza esta característica:

```
main( )
{
 int var1= 11;
 printf(" El contenido de Var1 es: %d",var1);
 printf(" La dirección de memoria donde está
Var1 es: %d", &var1);
}
```

Si el lector no comprende muy bien todavía cómo funciona exactamente el tema de las direcciones de memoria y relacionados, no debe preocuparse, ya que el tema de punteros y memoria es lo más complejo del lenguaje C y ya habrá tiempo en futuros capítulos para tratar el tema con profundidad.

### LA FUNCION GETCHE

Como se ha dicho ya, la función *getche()* está especializada en recoger una sola pulsación del teclado (un carácter) sin que el usuario necesite pulsar Enter para validar la entrada del dato, por lo que no sirve para la entrada de números u otros datos complejos.

Esta función tiene también muchas aplicaciones, como por ejemplo recoger la opción elegida entre un menú, donde sólo pulsando el número de opción el programa automáticamente podría ya ejecutar el código correspondiente.

## La parte relativa a las direcciones de memoria y los punteros es lo más difícil del lenguaje C

Ejemplo de un programa:

```
main( )
{
 char ch;
 printf("Escriba un caracter: ");
 ch = getch();
 printf(" El caracter introducido es:%c", ch);
}
```

La forma de funcionar de *getch()* es algo diferente a la de las dos anteriormente explicadas ya que no se le indica la variable destino como un parámetro, sino como un destino de la función. Ello se detallará más a fondo en próximos capítulos, ya que es algo muy usado también en C: recoger datos procedentes de funciones.

## Un paso adelante

**Con todo lo explicado hasta ahora el lector ya es capaz de poder definir variables en un programa C. Pero que podamos almacenar datos en el ordenador y se pueda operar con ellos no es muy útil por sí sólo. Por eso, además de lo anterior, tenemos que poder ser capaces de introducir datos nuevos durante la ejecución del programa y visualizar los valores resultantes de operaciones que realicemos en él.**

**Por todo lo dicho, lo primero que se va a explicar respecto a la programación en C es lo referente a las funciones (instrucciones) llamadas de Entrada/salida de datos.**

**Como instrucciones de salida de datos más conocidas tenemos la famosa printf(), que ya apareció en el primer programa de ejemplo del capítulo anterior, y como instrucciones de entrada de datos se van a explicar scanf() y getche(), la primera usada para poder entrar cualquier tipo de dato y la segunda especializada en captar pulsaciones de teclas por parte del usuario.**

CURSO DE ENSAMBLADOR

# Memoria y tipos de datos

**El ensamblador será nuestra tercera piedra de toque para establecer una sólida base de programación. En esta ocasión tendremos la oportunidad de echar un vistazo a temas como los caracteres ASCII o las variables en ensambladores.**

Seguimos estudiando las diferentes formas de almacenar los datos necesarios para nuestro trabajo.

### NUMEROS NEGATIVOS

Como se ha explicado, los bytes pueden almacenar en principio números en formato binario que equivalen al rango 0-255 en decimal, pero entonces alguno de los lectores se preguntará: "¿Y qué pasa entonces con los números negativos?. Como es lógico, el ordenador debe ser capaz también de poder operar con números negativos, ya que si no se verían muy limitadas sus capacidades de cálculo y en consecuencia su utilidad práctica.

**Agrupando bytes individuales en bloques nos lleva a obtener bloques de memoria que reciben nombres definidos**

Para solucionar el problema del signo tenemos que para poder representar números con signo se pasa a usar el último bit (el bit 7) para registrar si el número almacenado en él posee o no signo (bit 7=0 positivo, 1= negativo). Este sistema soluciona el problema, aunque también posee un inconveniente asociado, ya que el byte pierde rango de positivos representables, pasando de ser el número máximo representable +255 a ser +127, aunque por otro lado se gana por debajo (negativos), ya que ahora el rango mínimo pasa de ser 0 a –128. Tenemos así solucionado el problema del signo.

### AGRUPACIONES SUPERIORES AL BYTE

Si agrupamos bytes individuales en bloques de bytes obtenemos bloques de memoria que también poseen nombres estándares (aparte de las medidas de magnitud que se han explicado antes). Los bloques estándares son los siguientes:

- 2 bytes: Palabra.
- 4 bytes: Doble palabra.
- 8 bytes: Cuádruple palabra.
- 16 bytes: Párrafo.
- 256 bytes: Página.
- 64 Kbytes: Segmento (64*1024 bytes).

### EL TIPO DE DATO BCD

El 8086, además de los números en formato binario tanto con signo como sin signo propiamente dicho, soporta también un sistema de codificación de información llamado BCD.
Los datos BCD son un sistema de almacenamiento de decimales en formato binario y las siglas de su nombre provienen de las palabras inglesas *Binari Coded Decimal*, que se traduce literalmente como Decimal Codificado en Binario.
BCD básicamente es un sistema de representación de información (numérica) donde los dígitos del número decimal se almacenan con una equivalencia binaria directa, teniendo de esta forma que cada dígito requiere 4 bits para poder ser representado, y que son los bits mínimos que necesitamos para poder representar las 10 combinaciones de dígitos que tenemos en el sistema decimal.
A partir de este método de almacenamiento de números se derivan dos clases de BCD. Por un lado tenemos los llamados BCD's desempaquetados y por otro los BCD's empaquetados.

### BCD'S DESEMPAQUETADOS

Un número decimal está almacenado como un BCD desempaquetado cuando sólo se tiene almacenado un dígito de la cifra decimal por byte usado, quedando de esta forma que cada byte tiene sólo 4 bits con contenido útil (los 4 más bajos) que representan sólo un número de la cifra almacenada. Los 4 bits superiores de cada uno de los bytes usados para representarlo quedan con el valor 0 sin ningún contenido útil (por convenio, norma Intel). Como resumen tras lo dicho, queda claro que en consecuencia de este sistema, sólo podemos almacenar un valor del 0 al 9 por byte y para almacenar por ejemplo el número 2.360 necesitaremos 4 bytes de memoria (1 dígito = 1 byte).

### BCD'S EMPAQUETADOS

Un número decimal almacenado en formato BCD empaquetado difiere del anterior simplemente en que cada uno de los bytes usados para almacenarlo contiene dos dígitos decimales representados y no uno, como pasaba en el subtipo desempaquetado. Así pues, tenemos que los 4 bits superiores (*nibble* superior) contienen el número más significativo (en vez de estar a cero, claro) y los 4 inferiores contienen el dígito de menos valor del par que cabe en dicho byte.

**Binari Coded Decimal es un sistema de representación de información numérica**

Con este sistema, los números decimales BCD requieren la mitad de espacio necesario para ser almacenados (en bytes), respecto a los BCD desempaquetados, aunque en contrapartida se hace algo más difícil su manipulación posterior dado que no hay instrucciones de acceso directo a cuartetos individuales de bits dentro de un byte (los llamados *nibbles*), teniendo entonces que usar operaciones lógicas y/o rotaciones de bits.

### CARACTERES ASCII

La llamada tabla ASCII es un estándar americano de caracteres para el intercambio de información (*American Standar Code for Information Interchange*). Los bytes siempre almacenan números binarios, eso está claro, pero esos números binarios se pueden hacer que equivalgan a un conjunto determinado de caracteres, con lo que tenemos que podríamos representar 256 caracteres diferentes con cada byte, de forma que se podrían codificar textos en forma de cadenas de bytes.
Pero las equivalencias entre número y caracter correspondiente tienen que estar estandarizados para que la misma información equivalga en todos los ordenadores siempre a la misma *cadena de caracteres*, y por ello se creó el ASCII, que define a qué caracter equivale cada uno de los 128 primeros números (0-127), mientras que los 128 restantes son variables dependiendo del ordenador y del idioma que se instale en el mismo.

ESQUEMA DEL SISTEMA INTEL DE ALMACENAMIENTO DE INFORMACIÓN

EJEMPLO DATO TIPO WORD (2 bytes)

Representación teórica de un dato WORD

| byte B | byte A |
|---|---|
| 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |

bit

Dato tipo WORD almacenado en memoria

| byte A | byte B |
|---|---|
| 7 6 5 4 3 2 1 0 | 15 14 13 12 11 10 9 8 |

bit

EJEMPLO DATO TIPO DOUBLEWORD (4 bytes)

Representación teórica de los 4 bytes del dato

| byte D | byte C | byte B | byte A |
|---|---|---|---|
| 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |

bit

Dato 32 bits almacenado en memoria

| byte A | byte B | byte C | byte D |
|---|---|---|---|
| 7 6 5 4 3 2 1 0 | 15 14 13 12 11 10 9 8 | 23 22 21 20 19 18 17 16 | 31 30 29 28 27 26 25 24 |

bit

*Imagen de ejemplo del famoso Macro Assembler con un programa ensamblador abierto que va a ser ensamblado y convertido a EXE.*

El conjunto de caracteres ASCII incluye varios tipos:

- Letras mayúsculas y minúsculas.
- Los dígitos decimales.
- Caracteres especiales.
- Códigos de control y símbolos especiales.
- Símbolos de dibujado de recuadros.
- Símbolos de relleno y sombreado.
- Letras griegas.
- Símbolos científicos.
- Caracteres especiales de determinadas lenguas.

## VARIABLES EN ENSAMBLADOR

Como es fácil de comprender, si sólo se dispone de bytes individuales para poder almacenar números binarios de información, tendríamos que el ordenador sólo sería capaz de poder almacenar números sin signo del 0 al 255 y números con signo del -128 al +127, lo que haría del ordenador una máquina prácticamente inútil.

**Los microprocesadores de Intel poseen una regla o norma de almacenamiento común a todos**

Por lo dicho, los ordenadores se diseñaron para ser capaces de poder manejar bloques (variables) de 2 bytes (palabra), 4 (doble palabra) o hasta 8 bytes (cuádruple palabra: en los ordenadores más actuales y modernos) para poder así manejar números de mayor tamaño.

Trabajar con bloques superiores a un byte significa que, por ejemplo, si usamos un bloque de 2 bytes como una gran celda de memoria, el byte primero se usa como bits 0 a 7 del número que vayamos a almacenar en él, mientras que el segundo byte contendrá los bits 8 a 15 del número a almacenar, por lo que tenemos que en este caso se pueden almacenar números binarios de 16 bits.

En el 8086, que es el tema que se está estudiando en este curso, sólo se pueden usar paquetes de tamaño de 2 bytes, aunque ello es suficiente para poder representar números sin signo por ejemplo del 0 al 65535.

## COMO SE ALMACENAN LAS VARIABLES EN MEMORIA

Todos los microprocesadores de la casa Intel poseen una regla o norma de almacenamiento de información que se respeta siempre y que es usada en todos los tipos de datos sea cual sea su longitud, tanto en datos numéricos como en cadenas ASCII y cualquier tipo indistintamente. Esta norma es el llamado formato de almacenamiento de datos Intel (otros fabricantes pueden usar otros sistemas diferentes, por eso se especifica que es Intel), y su aplicación afecta al orden en el que se almacenan los bytes de datos en la memoria, ya que en los micros de la familia x86 Intel, cuando los datos están formados por 2 o más bytes de contenido se almacenan en orden inverso y no de forma secuencial de mayor a menor valor como sería de esperar por lógica tradicional.

Dicho lo anterior, cuando tenemos definida una variable de 16 bits (el llamado tipo *word*) en memoria, por ejemplo, que está formada por 2 bytes, el primer byte no contiene los bits 8 a 15 del número como sería de esperar, si no que contiene los bits 0 a 7, quedando por lo tanto los 8 bits más significativos (el byte alto) del número contenidos en la posición +1 respecto al inicio de la variable.

**Intel almacena los datos de dos o más bytes en sentido inverso y no secuencialmente como sería lógico**

Esta norma, que en la práctica no presenta ningún problema, se aplica a todos los niveles de datos, teniendo también que cuando representamos una variable de 32 bits por ejemplo (4 bytes), además de la inversión de orden de las dos parejas de bytes, las propias palabras que forman el número también están a su vez invertidas en orden, quedando entonces que tenemos los bytes en el siguiente orden:

Byte posición +0:byte 0, posición +1:byte 1, posición +2:byte 2, posición +3, byte 3.

Como se puede observar, con este método quedan todos los bytes en orden inverso respecto al original.

Aunque en operaciones normales todo ello es controlado automáticamente por la propia CPU, cuando nos encontremos en casos de manipulación de subcomponentes de una variable debemos tenerlo presente para no acceder a bytes erróneos en alguna ocasión. En el ejemplo que hay más adelante de una suma de dos componentes de un número de 32 bits se puede observar esta norma fijándonos en el orden en el que se suman las palabras de las variables.

## BIBLIOGRAFIA RELACIONADA


- *8088-8086/8087 Programación ENSAMBLADOR en entorno MS-DOS*. Miguel Ángel Rodríguez Roselló. Anaya Multimedia.
- *Programación del 80386/387. Manual de referencia y técnicas avanzadas de programación para diseñadores de sistemas, programadores y usuarios avanzados*. John H. Crawford, Patrick P. Gelsinger. Anaya Multimedia.
- *Programación del Z80*. Rodnay Zaks. Anaya Multimedia.
- *80286 Arquitectura y sistemas*. Grupo Waite, Edmund Strauss. Anaya Multimedia.
- *Nueva guía del programador en Ensamblador para IBM PC, XT, AT y compatibles*. Peter Norton, John Socha. Anaya Multimedia.

Autor: Hengomar

# Disco Fighter

**Decir que *Street Fighter* fue un juego que creó escuela no es nada nuevo. Muchos son los títulos que han seguido el mismo estilo. *Disco Fighter* es uno de ellos, pero no uno cualquiera. Es el juego creado con Div que ha resultado ganador de nuestro concurso.**

Hemos decidido ceder el espacio a su autor para que sea él mismo quien nos hable de los secretos de su Disco Fighter (¿quién mejor?). De modo que reproducimos el texto que el propio autor nos ha enviado:

En primer lugar, creo que debemos empezar por indicar los procesos que intervienen en el juego, ya que los más importantes de éstos se ejecutan simultáneamente durante el transcurso del programa, a saber: Control1p, Control2p y ControlIA, Cámara, Animación1 y Animación2, Sombra y sombra2, Especiales y especiales2 y IALurdes y MovimIA.

Control1p, Control2p y ControlIA. Éstos son los procesos encargados de mover a los personajes y realizar las acciones que le indique el jugador durante el juego. Control1p se ejecuta tanto en el modo de 2 jugadores como en el modo de 1 jugador, mientras que los otros dos son específicos de uno u otro modo de juego. Control2p se ejecuta sólo en el modo versus, contra otro jugador humano y ControlIA se ejecuta sólo en el modo arcade, contra el ordenador. Estos tres procesos son muy similares entre sí, variando tan sólo en unas cuantas cosas que los diferencian. Hay que decir también que el proceso ControlIA se ejecuta a la vez que otros dos, llamados Ialurdes y MovimIA. En conjunto, estos tres son los procesos que realizan las acciones del ordenador. ControlIA actúa igual que Control1p y control2p tan sólo recibe ordenes y las ejecuta, con la única diferencia de que ésas ordenes se las da otro proceso.







## Disco Fighter es el interesante juego de lucha que ha ganado nuestro concurso

Cámara. Este pequeño proceso es el encargado de mover el scroll y el modo7 del suelo.

Animación y animación2. Estos dos procesos son los encargados de realizar la animación de movimiento de los personajes. Permanecen a la espera de que el jugador dé la orden de movimiento y cuando lo hace proceden a animar el gráfico del luchador que tienen asignado: Animación, el primer jugador, Animación2, el segundo.

Sombra y Sombra2. Estos dos procesos son los encargados de poner en pantalla y animar las sombras de los luchadores, que repetirán todos los movimientos que estos realicen; para ello, simplemente "observan" qué gráfico tiene asignado el jugador correspondiente y se lo asignan espejado a sí mismo. Si coincide que los dos luchadores son el mismo personaje, estos procesos se encargan de cambiarle el color al segundo jugador para que sean perfectamente distinguibles.

Especiales y especiales2. Controlan si los jugadores que tienen asignados realizan algún movimiento especial: una onda o un movimiento de carga. Para ello, llevan cuenta de las teclas pulsadas y si el jugador realiza alguno de los dos movimientos posibles pasan a ejecutar el proceso correspondiente.

IaLurdes y movimIA. IALurdes realiza, aleatoriamente, uno de los varios movimientos posibles y MovimIA es la encargada de enviar según esos datos las órdenes correspondientes al proceso ControlIA para que éste las lleve a cabo definitivamente.




Una vez que hemos conocido cúal es la función de cada proceso, podemos entrar en más detalle explicándolos paso por paso.

### CONTROL DE LOS PERSONAJES

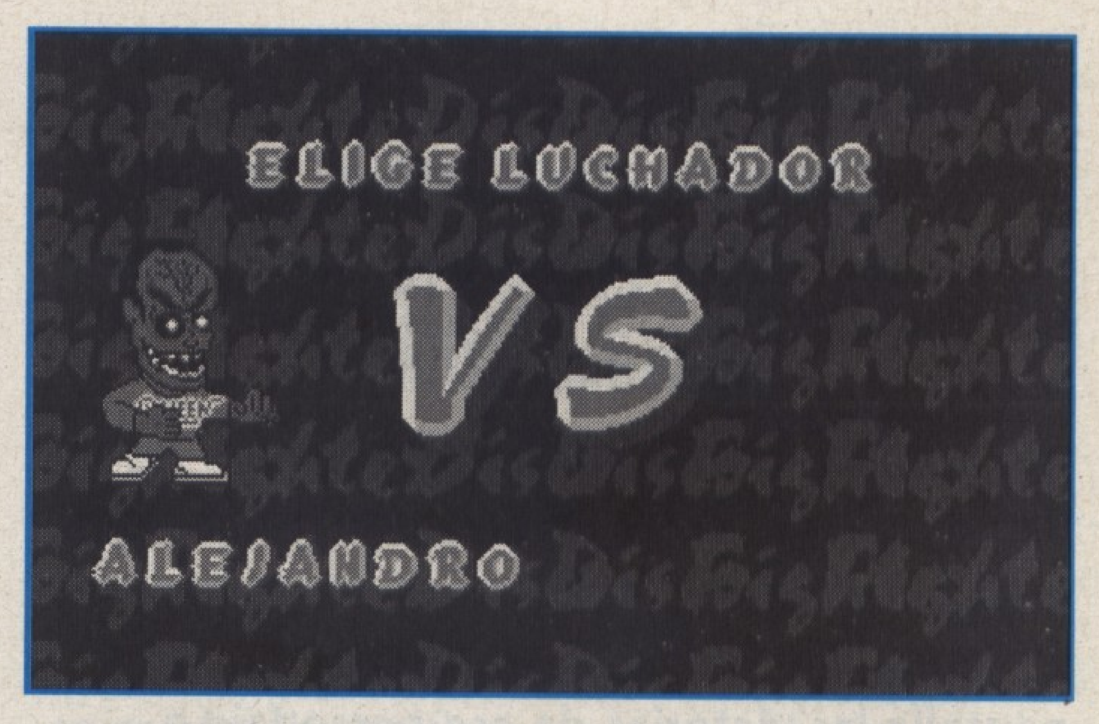
Como ya se dijo antes, tres son los procesos encargados de llevar a cabo esta tarea, aunque sólo dos se ejecutarán simultáneamente dependiendo del modo de juego. El primero de ellos es Control1p, el encargado del control del primer jugador.

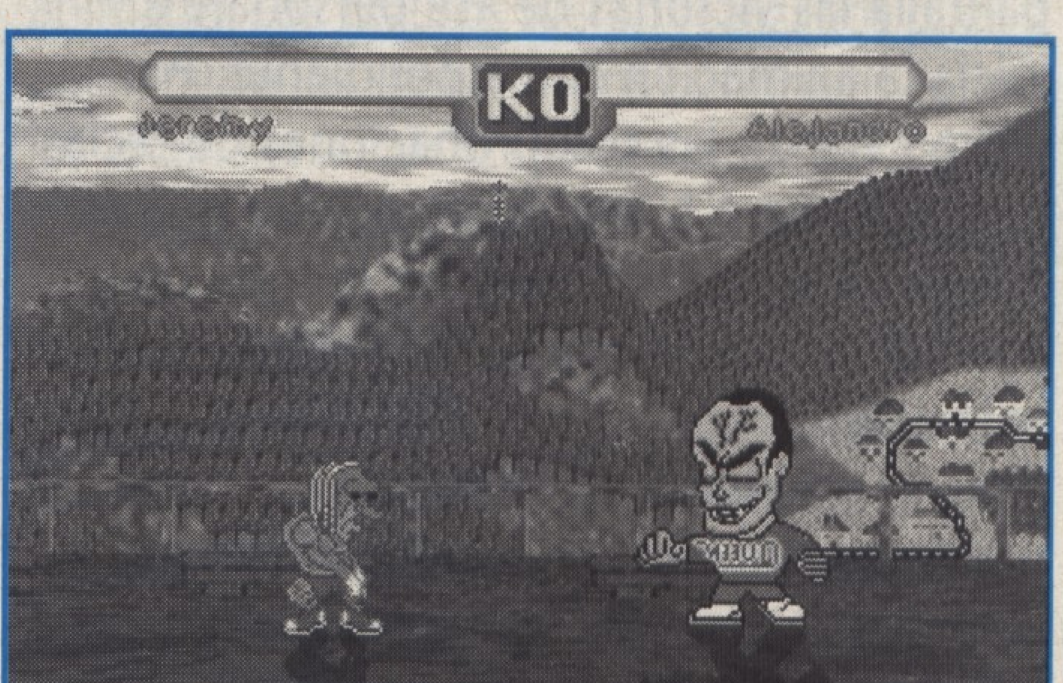
Este proceso comienza indicando el archivo del que se usarán los gráficos, en este caso es xogador1. A continuación, inicializa algunas variables e indica las coordenadas iniciales del personaje fijando su x y su y. Llama también a los procesos sombra, marcador y marcador2. Después de hacer esto por varias sentencias "if", comprueba cuál es el luchador seleccionado y, una vez encontrado, pasa a realizar el movimiento de presentación de ese luchador antes del combate. A continuación, detiene la ejecución del proceso durante unos segundos por la sentencia frame(36000) mientras aparecen y desaparecen los "letreros" de combate. Por fin, entramos en el bucle principal, el que realiza el movimiento en sí. Uno de los primeros bucles vigila; si no se ha pulsado ninguna tecla de desplazamiento lateral, izquierda o derecha, y la variable local de movimiento de ese personaje vale cero, pasa a realizar la animación del personaje cuando éste se encuentra sin hacer ninguna acción.

```
if(not(key(_right)) and not(key(_left))and
andando==0)
        if (timer[1]>=5)

graph=anim0[paso++];timer[1]=0;
        end;
        ...
```




El timer controla el tiempo necesario para cambiar de frame y luego actualiza el gráfico conforme a los valores almacenados en el vector anim0. Por último, vuelve a inicializar el contador asignándole el valor cero. A continuación se comprueba si se ha llegado al final del vector anim0; si es así, se le asigna de nuevo el valor cero a la variable paso.

```
IF (paso==sizeof(anim0))
        paso=0;
        graph=2;
    end
```

Por último se asigna el gráfico: graph=anim0[paso].

## Hay tres procesos que se encargan de controlar a los personajes

Después aparece un pequeño bucle "if (golpe2<>nada)" que comprueba si la variable golpe2 tiene un valor distinto de la constante "nada"; si es así, es que se ha producido un golpe por parte del adversario, por lo tanto realiza la animación de golpeado.

Otro bucle es "if (esp==ondas)". Lo que hace es comprobar si la variable que guarda el valor para los movimientos especiales tiene el valor "ondas", lo que indicaría que el primer jugador ha realizado el movimiento correspondiente. Así que anima debidamente al personaje y una vez hecho esto devuelve a la variable "esp" el valor cero. Luego pasa a comprobar las coordenadas "x" correspondientes a los luchadores para así colocar el gráfico de cada proceso debidamente, es decir mirándose uno al otro. Con esto se evita que, al intercambiar sus posiciones, los luchadores aparezcan de espaldas. Se comprueba también si los luchadores están lo suficientemente cerca como para considerar que ya no se pueden acercar más; si esto es así, la variable "contacto" se pone a uno con el fin de indicarlo; así, al comprobar las teclas de movimiento también se comprueba que esta variable posea el valor cero, indispensable para entrar en el bucle y que, por lo tanto, el personaje se mueva. Estás comprobaciones se realizan según el valor que tome *flags*, porque así sabemos también qué tecla es la que debe pulsar el jugador para cubrirse, ya que ésta cambia según hacia donde mire el personaje. Si se pulsa dicha tecla, el valor de la variable cubrir1 se cambia a uno.

A continuación se pasa a comprobar por medio de sentencias "if" consecutivas si se pulsan las teclas correspondientes a los puñetazos y patadas. Si se pulsa alguna, se indica el sonido y se "duerme" el proceso de animación para que no interfiera con la nueva animación que debe llevar a cabo ahora.

Después, se comprueba si el gráfico colisiona con el otro luchador; si es así y la variable "cubrirse" del otro personaje está a cero, se indica el golpe que se ha efectuado y se llama al proceso "quita_enerxía", que será el encargado de mover la barra de energía del personaje golpeado. Si el personaje golpeado estaba cubriéndose, entonces es la sentencia "else" la que se lleva a cabo. Duerme el proceso de animación del segundo personaje y congela a éste para realizar la animación correspondiente sin interferencias; en cuanto termina vuelve a despertar a los anteriores procesos, pero como también hay que comprobar si se colisiona con el proceso de control del ordenador, se hacen de nuevo las comprobaciones anteriores si se detecta una colisión. Todo esto se hace con todos los movimientos de ataque posibles, es decir, con los dos puñetazos y con las dos patadas.

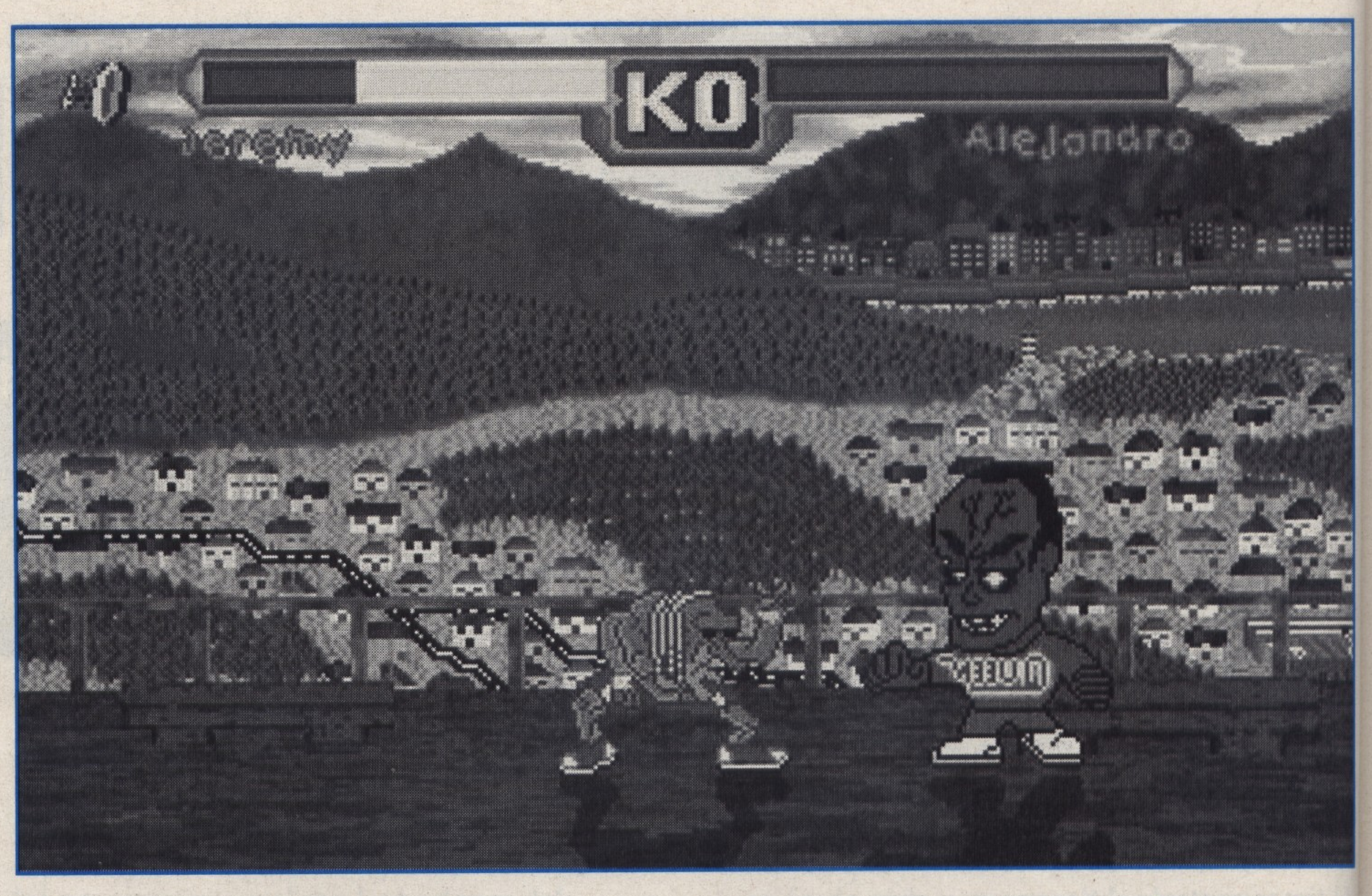
Mas abajo, en el código, encontramos la parte que hace referencia a los saltos y a los movimientos de ataque que se pueden producir en el aire. Para realizar estos golpes mientras se está saltando se utiliza una variable y un timer. La variable "anterior" indica, cuando vale uno, que ya se ha realizado un ataque en el aire. Y el contador sirve para saber cuánto tiempo hace que se produjo. Sólo se puede hacer un movimiento por salto, por lo tanto si la variable "anterior" vale uno ya no se entra en el bucle. Si entra, el contador se incrementa para evitar que el personaje reste energía de más a su contrincante.

Antes vimos cómo el proceso comprobaba si el jugador lanzaba una onda, pero no vimos cómo hacía lo mismo con el otro movimiento especial. Éste se comprueba a continuación. Si la variable "esp" toma el valor "lanz" es que se ha producido dicho movimiento especial por lo que entra en el bucle y lo ejecuta. Como este movimiento es directamente ejecutado por el personaje, es decir, es él en persona quien lo lleva a cabo, decidí incluirlo dentro del código principal del mismo, pero se podría haber creado perfectamente como un proceso aparte; sin embargo, incluyéndolo en el proceso control1p me evitaba ciertos problemas con los procesos de sombras. El desarrollo interno del bucle de este movimiento sigue un procedimiento muy similar al del propio control1p: primero se comprueba qué luchador es, para asignarle así los gráficos, y después se le desplaza y comprueba si ha colisionado contra su rival. Hay que decir

que los movimientos especiales contienen una variable denominada "activo" que indica, cuando contiene el valor uno, que en ese momento se está realizando un movimiento de ese tipo.
A continuación se comprueba el valor de la variable "K.O.". Ésta, si vale uno, indica que ese personaje está K.O., es decir, se le agotó su barra de energía. Si es así, indica en pantalla las victorias de ese personaje e inicia la animación correspondiente. Por último vienen las comprobaciones de teclas de golpe, pero teniendo en cuenta que se pulsan simultáneamente con la tecla de agacharse, por lo que tendrán otra animación.
Todo lo anteriormente explicado es prácticamente idéntico para los procesos control2p y controlIA, con la salvedad de que el último recibe las ordenes a través de otros dos procesos "MovimIA" y "IALurdes".

**MODO ARCADE**

Explicaremos ahora el método usado en este juego para el modo de lucha contra el ordenador, analizando como hace éste para moverse por sí sólo.
El proceso IALurdes posee varios vectores donde tiene almacenados unos datos; éstos hacen referencia a un movimiento concreto y en conjunto logran un movimiento complejo del personaje. El nombre de cada uno describe superficialmente lo que realiza ese vector: huida, huida2, saltobaj... Después de una pausa comienza el bucle que elegirá aleatoriamente un número del uno al nueve, ambos inclusive. Si es distinto de uno, entra en una sentencia *switch* que va leyendo el vector correspondiente al movimiento y se lo asigna a la variable "acción". Es aquí cuando entra en escena el proceso MovimIA. Como esté está ejecutándose todo el tiempo también sabe en todo momento el valor que Ialurdes está leyendo del vector.; de modo que éste va entrando en los "case" de su sentencia *switch* a medida que éstos van cambiando su valor. Pero además de esto también comprueba la distancia al otro personaje para actuar en consecuencia aleatoriamente.




**SOMBRAS**

Una de las cosas que siempre estuvo muy descuidada en los juegos de lucha fueron las sombras. Sin duda recordaréis como éstas eran siempre unas manchas oscuras en el suelo que seguían al personaje y, en algunos casos, incluso tenían la desafortunada propiedad del parpadeo. Tan sólo el famoso *Killer Instinct* tenía sombras reales. Partiendo de esta base, siempre tuvimos en mente que Disco Fighter también las tendría como las tienen ahora muchos juegos de lucha en 3D.
Lo primero que hacen estos dos procesos es cargar de nuevo el fpg correspondiente al luchador que tienen asignado. Luego transforman la paleta de dicho fpg al color de la paleta que posee un negro, que será el utilizado para las sombras. Con esto se logra que absolutamente todos los gráficos contenidos en dicho archivo se vuelvan negros. Después, simplemente se entra en un bucle que asigna a la variable local "graph" de este proceso el valor que tiene esa misma variable del proceso padre. Lo mismo hace con las variables "flags" y "x". Acto seguido comprueba que el valor de la "y" del proceso padre sea mayor de 150; si no es así pasa a incrementar su variable local "y" para que ésta realice el movimiento contrario al proceso padre.

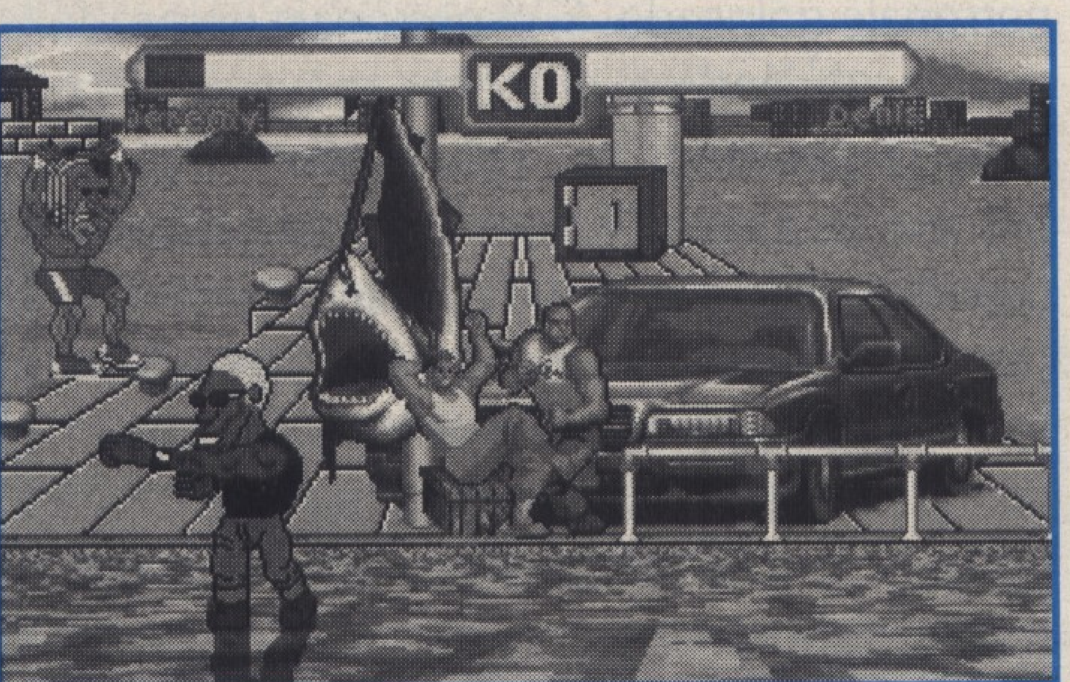
**Para realizar un golpe en el aire, se usa la variable "anterior" y un timer**

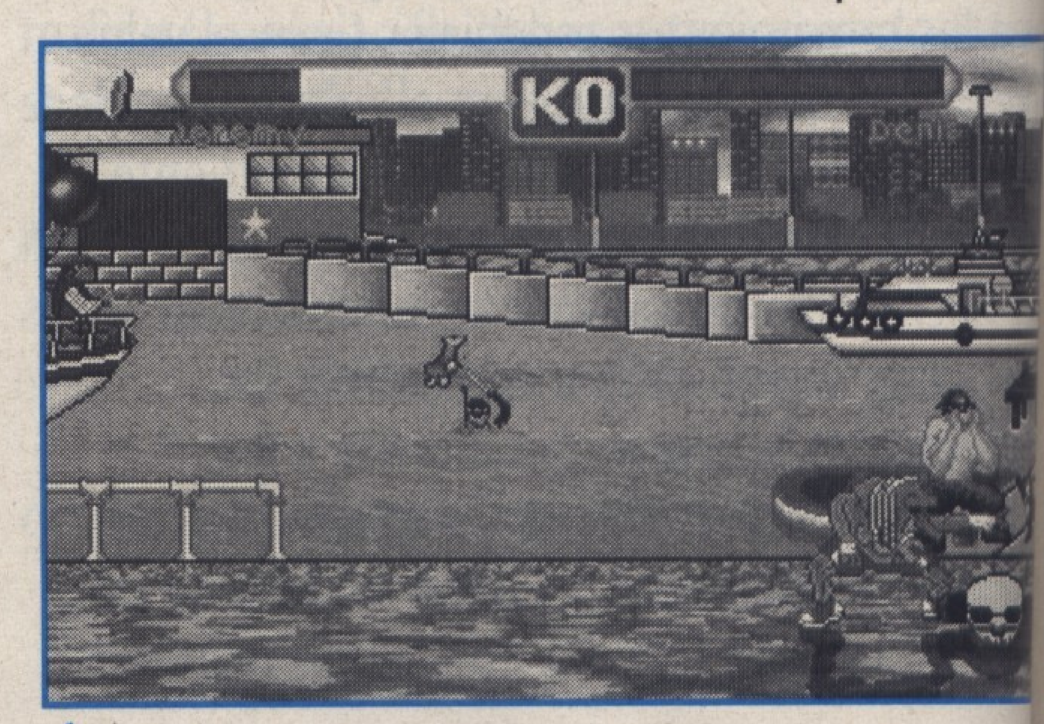
Termina asignándole un valor para evitar problemas.
El proceso sombra2 es idéntico salvo por una sentencia "if", la que comprueba si los dos luchadores son iguales. Si efectivamente son iguales pasa a asignarle el valor correspondiente al personaje. A continuación entra en el bucle correspondiente y transforma su paleta a la indicada en el *offset*. Dichos valores los podréis ver en la sección de declaración de globales, al principio del programa.

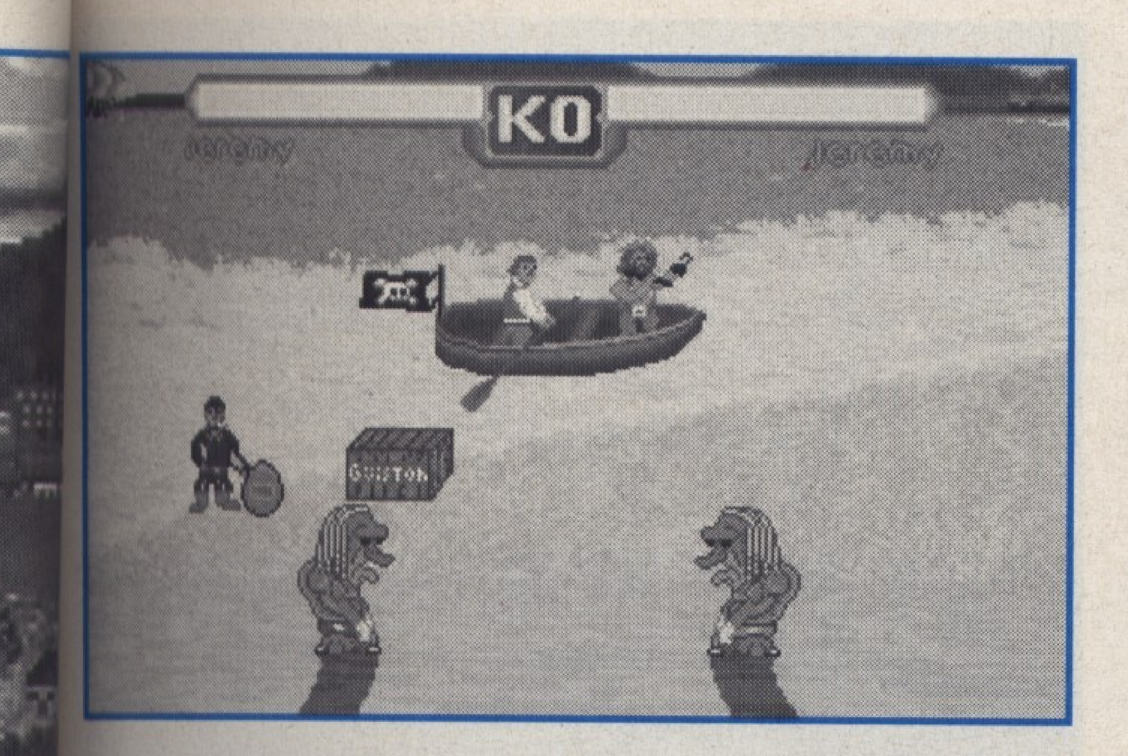
**MOVIMIENTOS ESPECIALES**

En estos dos procesos son necesarias cuatro variables. Lo primero que hacemos es ponerle una prioridad alta a este proceso para que así se ejecute antes de cualquier otro. El porqué de esto es sencillo: de no hacerlo así nos encontraríamos con que, de vez en cuando, el proceso de control se ejecutaría antes y el personaje realizaría un golpe, como un puñetazo, justo antes del movimiento especial correspondiente.
Después, inicializamos un contador a cero y entramos en el bucle donde se comprueba si






alguno de los dos luchadores está K.O. para actuar en consecuencia, actualizando las variables necesarias que serán leídas por otros procesos. A continuación, si el *timer* pasa de las 30 centésimas, se ponen a cero las primeras variables de los movimientos especiales. Luego se comprueba si se pulsa la tecla correspondiente a agacharse, tecla con la que comienzan los dos especiales posibles. Si se está pulsando, se activan las primeras variables de los dos movimientos. Ahora debemos seguir leyendo el teclado según las "flags" de los personajes, con lo que entraremos en el *case* correspondiente. Si se pulsa alguna tecla que sigue el movimiento correcto (por orden), la variable de movimiento anterior está activada y no se ha sobrepasado el tiempo de 30 centésimas, se pasa a incrementar el valor de esa variable al paso dos y el del otro especial a cero, ya que no se está ejecutando ese movimiento. Por último, se comprueba que se pulsa la tecla de golpe permitida, que las variables de especiales están al valor dos y que no hay ningún otro especial activo en ese momento. Si se cumplen todas estas condiciones se ha realizado correctamente un movimiento especial, se pasa a llamarlo y a actualizar las variables "esp" y "activo".

## VARIOS IDIOMAS

Una de las cosas que siempre me gusta poner en los juegos es la posibilidad de elegir el idioma en el que los textos aparecerán en pantalla. Así, el jugador tendrá la posibilidad de jugar en una lengua que entiende o que, por lo menos, le es más próxima, sin tener que conocer la lengua del programador.

Para ello lo primero que hice fue poner todos los mensajes que aparecerían en pantalla en un vector llamado "textos". Luego, según el lenguaje escogido, el proceso de opciones incrementaba o decrecía la variable "idioma". Una vez llegado al punto donde hay que escribir los textos, tan sólo es necesario indicar el vector donde están almacenados y su posición. Como los mensajes están dispuestos en tres grupos de diez elementos, se debe dar la siguiente orden de lectura del vector "textos[n+idioma]", donde n es el número que ocupa el texto en el vector y idioma la variable




que marca la lengua elegida. Por ejemplo, si queremos escribir "salir" debemos ver cuál es su posición, en este caso la tercera; por lo tanto, "textos[3+idioma]" valiendo cero la variable idioma, será "salir" en gallego (sair). Valiendo 10 será en castellano y, por último, valiendo 20 será en inglés (exit).

## PROCESO QUITA_ENERXIA

Es el proceso al que se llama cada vez que un luchador ha golpeado a otro. Para entrar en él, no debe de estar restando energía en ese momento, como indica la variable "quita" cuando vale cero, y el combate no debe haber terminado (termino=0). En la sentencia "switch" se busca el golpe que se ha efectuado y, una vez encontrado, se tiene en cuenta quien lo llamó, lo que viene dado por "father". De este modo sabemos a quien debemos restar la energía. Una vez hecho esto se entra en un bucle que, simplemente, desplaza la barra de energía tal y como indica el golpe efectuado.

## GRAFICOS Y SONIDO

Para la realización y diseño de los gráficos se comenzó utilizando el programa Deluxe Paint II enhanced que, aunque es bastante antiguo, cumple perfectamente. Se usó este programa porque los gráficos habían comenzado a crearse antes que DIV apareciese en el mercado. La idea para los gráficos era hacerlos tipo *Street Fighter II*, es decir, tipo dibujo animado. Los gráficos fueron realizados pixel a pixel, tanto los sprites de los personajes como los escenarios. Prácticamente todos los gráficos del juego son originales, aunque también se usaron capturas de otros juegos que fueron retocados para éste.

**Disco Fighter se presenta en varios idiomas para evitar los típicos problemas a la hora de entenderse**

Para los escenarios, la idea era usar lugares reales. Luego se empezó dibujando los contornos de los mismos y su esqueleto para, posteriormente, ir rellenándolos y completándolos poco a poco con muchos elementos distintos, desde iconos hasta selecciones de pantallas capturadas; es decir, varios de los elementos de los escenarios estaban hechos previamente a la realización de los mismos .

Algunos de los sprites fueron hechos en papel antes de pasar a dibujarlos con el ordenador para así tener una idea de referencia de lo que se iba a hacer. Esto también se usó en algunos de los escenarios que fueron dibujados a modo de boceto en el propio lugar en el que se inspiraban.

En cuanto al sonido, elegimos la música usando diversas canciones sacadas de distinta procedencia pero que guardaban alguna relación con el ambiente que le queríamos dar a cada escenario. Para las voces tuvimos que hacer uso del micrófono para digitalizarlas nosotros mismos y luego eliminar, con el cooledit, el ruido de fondo.



## JUEGOS GANADORES: 2º

Autor: Emilio Galindo
Jacob Almagro

# PA-PÚ

**El segundo juego ganador de este número es un adictivo programa, versión de un clásico que hemos podido disfrutar en diversas versiones y en distintas plataformas. Se trata de Pa-pú, un título que asegura jugabilidad con unos gráficos simpáticos y un sonido ameno.**

La breve historia que han ideado estos chicos gira alrededor de un planeta cabezoide gobernado por el rey Cabeza. Tenemos que salir de este planeta saliendo victoriosos de los combates de Pa-pú. A continuación os mostramos los controles que tenemos que manejar para jugar con este programa.

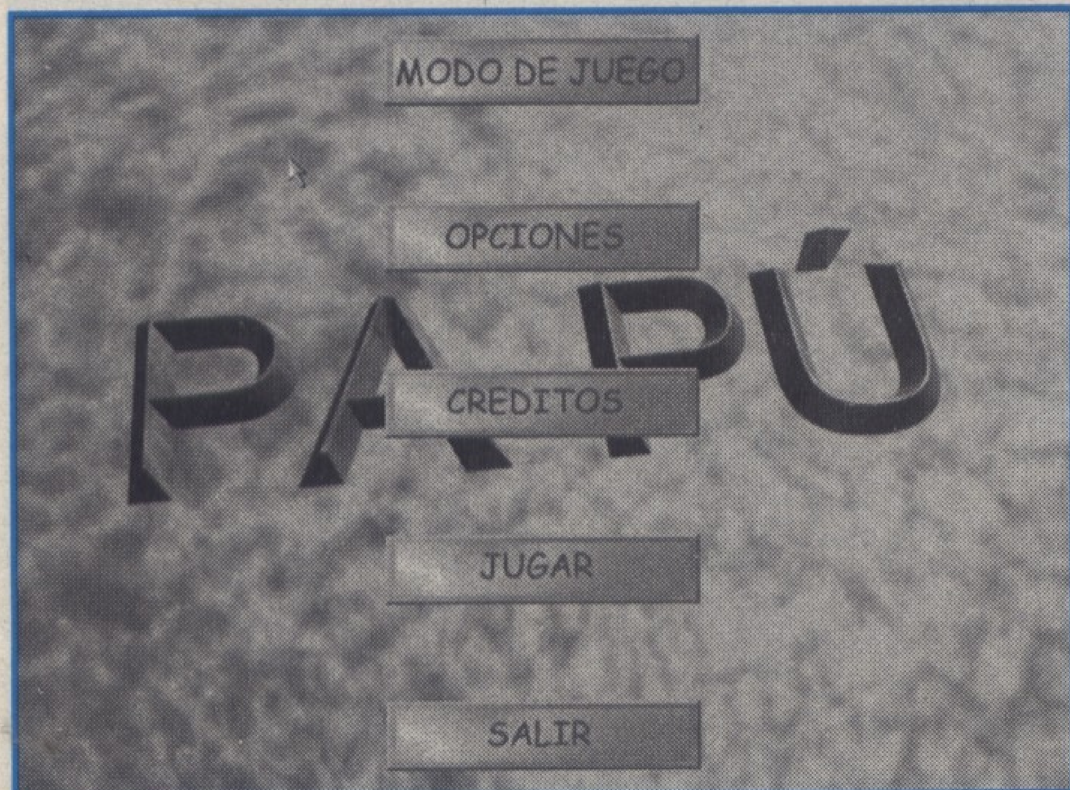
PAUSA / INTER: Detiene el juego.
ESC: Sales de los menús.
RATÓN: Te mueves por los menús.

Jugador 1:
W: Arriba.
S: Abajo.
D: Disparo de pulsómetros interespaciales (o algo así...).

Jugador 2:

TECLAS CURSOR: Arriba y abajo.




CTRL: Disparo de pulsómetros interespaciales (o algo así...).

El código que os mostramos en el siguiente listado no es el completo del juego, que encontraréis en el interior del CD-Rom, pero muestra algunos de los procesos más interesantes de este juego. Esperamos que os guste.

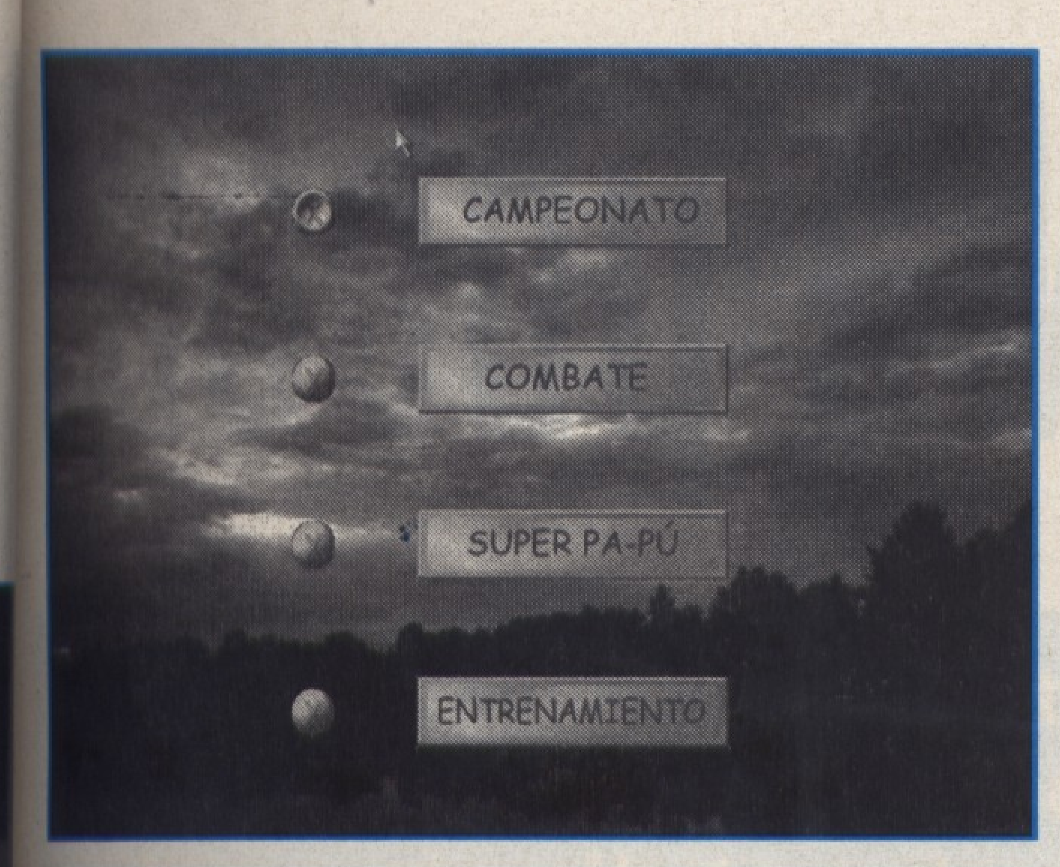
```
BEGIN        //Este programa principal sólo
carga ficheros y sonidos y llama
             //a la presentacion
SET_FPS(60,0);
SET_MODE(M640X480);
FICHEMENU=LOAD_FPG("MENUPA-P.FPG");
LOAD_FNT("NUEVOFNT.FNT");
CREDIT=LOAD_MAP("CREDIT.MAP");
SONI=LOAD_PCM("PRESENT.PCM",0);
PRESENT=LOAD_MAP("PRESENT.MAP");
MODO2GRA=LOAD_MAP("MODO.MAP");
INTRO();
END

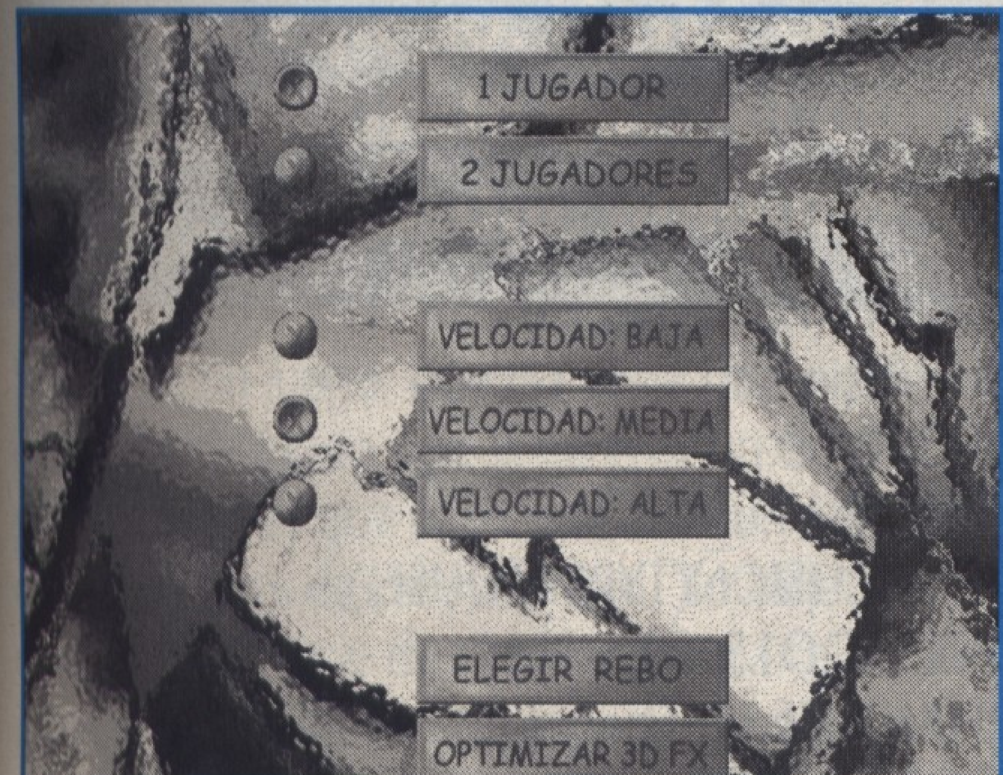
PROCESS INTRO();      //Presentación
BEGIN
SET_FPS(100,0);
LOAD_PAL("PRESENT.MAP");
FADE(200,200,200,8);
WHILE (FADING) FRAME; END
SOUND(soni,256,256);
put_screen(0,present);
FADE_ON();
WHILE (FADING) FRAME; END
WHILE(NOT KEY(_ESC) AND NOT KEY
(_SPACE) AND CONTADOR<200)
FRAME;
CONTADOR++;
END
CLEAR_SCREEN();
INTRO2();
END
```




```
PROCESS INTRO2();      //Segunda
presentación
BEGIN
UNLOAD_MAP(PRESENT);
UNLOAD_PCM(SONI);
SET_FPS(40,0);
LOAD_PAL("PRESENT2.MAP");
PRESENT2=LOAD_MAP("PRESENT2.MAP");
SONI2=LOAD_PCM("PRESENT2.PCM",0);
FADE(200,200,200,8);
WHILE(FADING) FRAME; END
SOUND(SONI2,256,356);
PUT_SCREEN(0,PRESENT2);
FADE_ON();
WHILE (FADING) FRAME; END
WHILE (NOT KEY(_ESC) AND NOT KEY
(_SPACE) AND NOT (MOUSE.LEFT) AND
CONTADOR2<200)
FRAME;
CONTADOR++;
END
LATINOR=LOAD_PCM("LATINO1.PCM",0);
SOUND(LATINOR,256,256);
SDISPARO=LOAD_PCM("DISPARO.PCM",0);
SBONUSL=LOAD_PCM("BUIU.PCM",0);
SBONUSC=LOAD_PCM("ACCION.PCM",0);
FUENTECOMBATE=LOAD_FNT("COMB.FNT");
FUENTECOMBATE2=LOAD_FNT("COMB22.FN
T");
LOAD_FPG("PA-PU2.FPG");
MEN=MENU();
END

PROCESS MENU();   //Menú, cada botón es un
proceso
BEGIN
LET_ME_ALONE();
CHOQUE=2;
CHOQUE22=2;
CHOQUEC=2;
```







```
DELETE_TEXT(ALL_TEXT);
UNLOAD_MAP(COMBMAP);
UNLOAD_MAP(PAPUMAP);
UNLOAD_MAP(ENTREMAP);
UNLOAD_MAP(PRESENT2);
UNLOAD_PCM(SONI2);
UNLOAD_PCM(PACO);
UNLOAD_MAP("ENTRENAMIENTO.MAP");
CONT2=0;   // Contadores de uso general
CONT3=0;
CONT4=0;
CONT5=0;
MARC1=0;
MARC2=0;
NPART=1; //Número de partidos (modo
campeonato)
PUNTSCRAT=0; //Puntos de cada palo
PUNTENDE=0;
PUNTPRED=0;
LOAD_PAL("MENUPA-P.FPG");
PUT_SCREEN(0,3);
MOUSE.GRAPH=1;
MOUSE.FILE=0;
FRAME;
LOOP   //Llama a los procesos de los botones
MODOO=MODO();
OPCIONE=OPCIONES();
CREDITO=CREDITOS();
JUGARR=JUGAR();
SALI=SALIR();
FRAME;
END
END

PROCESS MODO(); //Botón modo, los
procesos de botones tienen la misma
                        //estructura
BEGIN
GRAPH=PULSADO;
X=320;
Y=40;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO=21;  //Gráfico con el botón pulsado
IF(MOUSE.LEFT);
CONT2=0;
WHILE(CONT2<4); //Apaga y enciende
("suaviza") la pantalla
FADE_OFF();
FRAME;
CONT2++;
END
FADE_ON();
MODO2();
END
ELSE
PULSADO=20;  //Gráfico con el boton normal
END
END


PROCESS OPCIONES(); //Botón opciones
BEGIN
GRAPH=PULSADO2;
X=320;
Y=140;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO2=23;
IF(MOUSE.LEFT);
CONT2=0;
WHILE(CONT2<4);
FADE_OFF();
FRAME;
```




```
CONT2++;
END
FADE_ON();
OPCIO2=OPCIONES2();
END
ELSE
PULSADO2=22;
END
END

PROCESS OPCIONES2(); //Pantalla de
opciones
BEGIN
LET_ME_ALONE();
FONDOOPC=LOAD_MAP("FONDOHIELO.MAP
");
PUT_SCREEN(0,FONDOOPC);
LOOP
IF(KEY(_ESC) OR (KEY(_SPACE)));
CONT2=0;
WHILE(CONT2<4);
FADE_OFF();
FRAME;
CONT2++;
END
```




```
FADE_ON();
MENU();
BREAK;
END
FRAME;

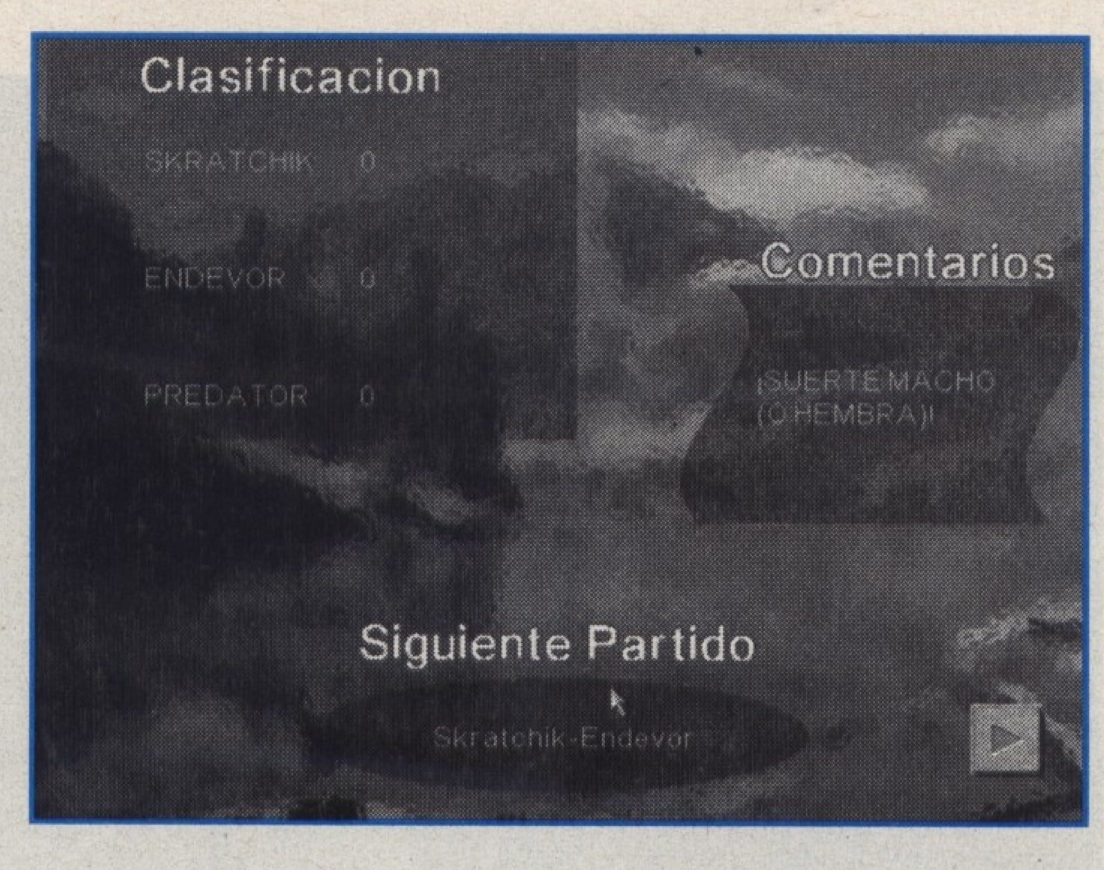

PLA1=PLAYER1(); //Se identifican los procesos
de los botones de la pantalla
PLA2=PLAYER2();
VELO1=VELOCIDAD1();
VELO2=VELOCIDAD2();
VELO3=VELOCIDAD3();
ELEREBO=ELEGREBO();
OPTIMI=OPTIMIZAR();

BOLAR1=BOLARO1();
BOLAR2=BOLARO2();
BOLAR3=BOLARO3();
BOLAR4=BOLARO4();
BOLAR5=BOLARO5();


END
END


PROCESS PLAYER1(); //Botón jugador1
BEGIN
GRAPH=PULSADO10;
X=380;
Y=50;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO10=5;
IF(MOUSE.LEFT);
BOLARP1=35;
BOLARP2=34;
CUANTOS=1;
END
ELSE
PULSADO10=4;
FRAME;
END
END


PROCESS PLAYER2(); // Botón jugador 2
BEGIN
```

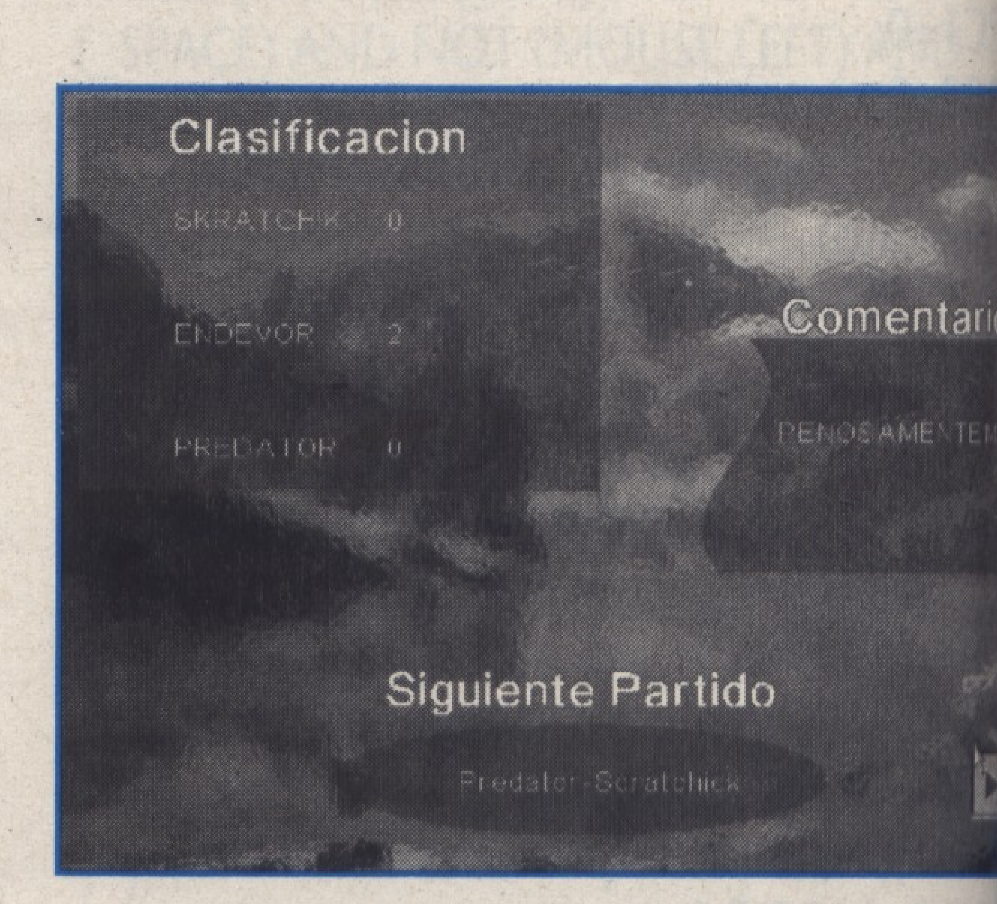



```
GRAPH=PULSADO11;
X=380;
Y=100;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO11=6;
IF(MOUSE.LEFT);
BOLARP1=34;
BOLARP2=35;
CUANTOS=2;
END
ELSE
PULSADO11=7;
FRAME;
END
END

PROCESS VELOCIDAD1(); // Velocidad baja
BEGIN
GRAPH=PULSADO13;
X=380;
Y=200;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO13=29;
IF(MOUSE.LEFT);
CORBOLX=10;
CORBOLY=10;
BOLARP3=35;
BOLARP4=34;
BOLARP5=34;
VELO=1;
END
ELSE
PULSADO13=28;
FRAME;
END
END


PROCESS VELOCIDAD2(); //Velocidad media
BEGIN
GRAPH=PULSADO14;
X=380;
Y=250;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO14=31;
IF(MOUSE.LEFT);
CORBOLX=15;
CORBOLY=15;
BOLARP3=34;
BOLARP4=35;
BOLARP5=34;
VELO=2;
END
ELSE
PULSADO14=30;
FRAME;
END
END


PROCESS VELOCIDAD3(); //Velocidad alta
BEGIN
GRAPH=PULSADO15;
X=380;
Y=300;
FRAME;
IF(COLLISION(TYPE MOUSE));
PULSADO15=33;
IF(MOUSE.LEFT);
CORBOLX=20;
CORBOLY=20;
BOLARP3=34;
BOLARP4=34;
BOLARP5=35;
VELO=3;
END
ELSE
PULSADO15=32;
FRAME;
END
END
```



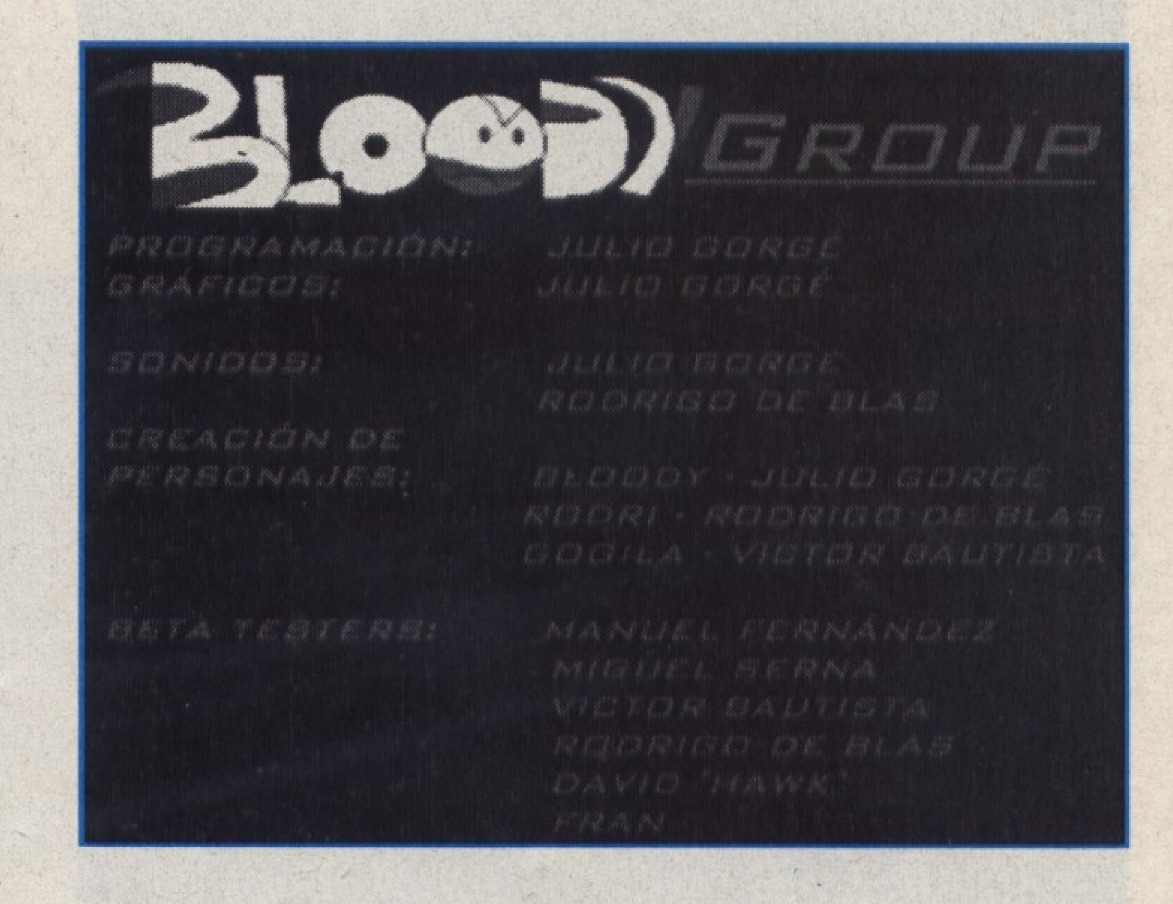



Autor: Julio Gorgé Frochoso

## JUEGOS GANADORES: 3º

# Bloddy's War '99

**En tercer lugar os presentamos un programa hecho exclusivamente para ser jugado por varios jugadores en un mismo equipo. Hasta cuatro personas podrán enfrentarse en un título para el que hacen falta buenas dosis de picardía, agilidad y, sobre todo, paciencia.**




Se trata de un juego que no tiene una pizca de acción, de modo que si no eres un tío paciente tendrás que acercarte a alguno de los otros ganadores de este número. Tu objetivo es recoger una serie de objetos por pantalla. El juego se desarrolla en una especie de subsuelo por el que te mueves gracias a un taladro, que funciona de distinto modo según cuál sea el terreno que tenga que atravesar.

En la pantalla principal del juego nos vamos a encontrar los siguientes menús:
JUGAR: Inicia una partida para 2 o más jugadores (no es en red).
OPCIONES: Muestra las opciones de juego.
SALIR: Sale del juego, por supuesto.
CÓMO JUGAR: Muestra un mapa del teclado, así como una ayuda.

A continuación incluimos parte del código, que será introducido en toda su extensión en el interior del CD-Rom que acompaña a la revista. Se trata de uno de los procesos más interesantes de este curioso juego.

```
PROCESS PLAYER(p);
PRIVATE
 graph1,graph2,graph3,graph4;
 vel;
 CX,CY;
BEGIN

file=f_players;

z=-250;
power=4;

IF(p==1);
  graph1=1;
  graph2=2;
  graph3=3;
  graph4=4;
  cx=1;cy=8;
END

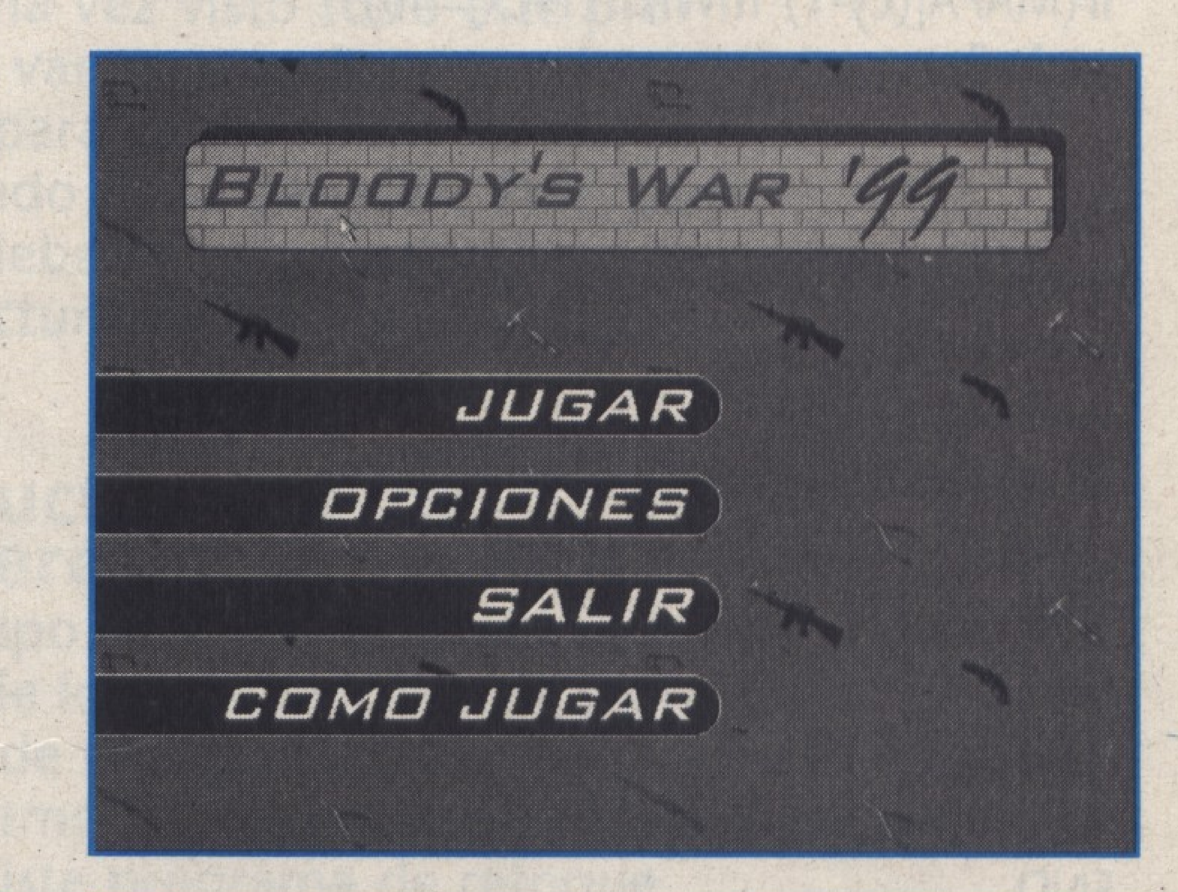
IF(p==2);
  graph1=5;
  graph2=6;
  graph3=7;
  graph4=8;
  cx=62;cy=46;
END

IF(p==3);
  graph1=9;
  graph2=10;
  graph3=11;
  graph4=12;
  cx=1;cy=46;
END

IF(p==4);
  graph1=13;
  graph2=14;
  graph3=15;
  graph4=16;
  cx=62;cy=8;
END

puedo2[p]=1;
arma_activa[p]=0;

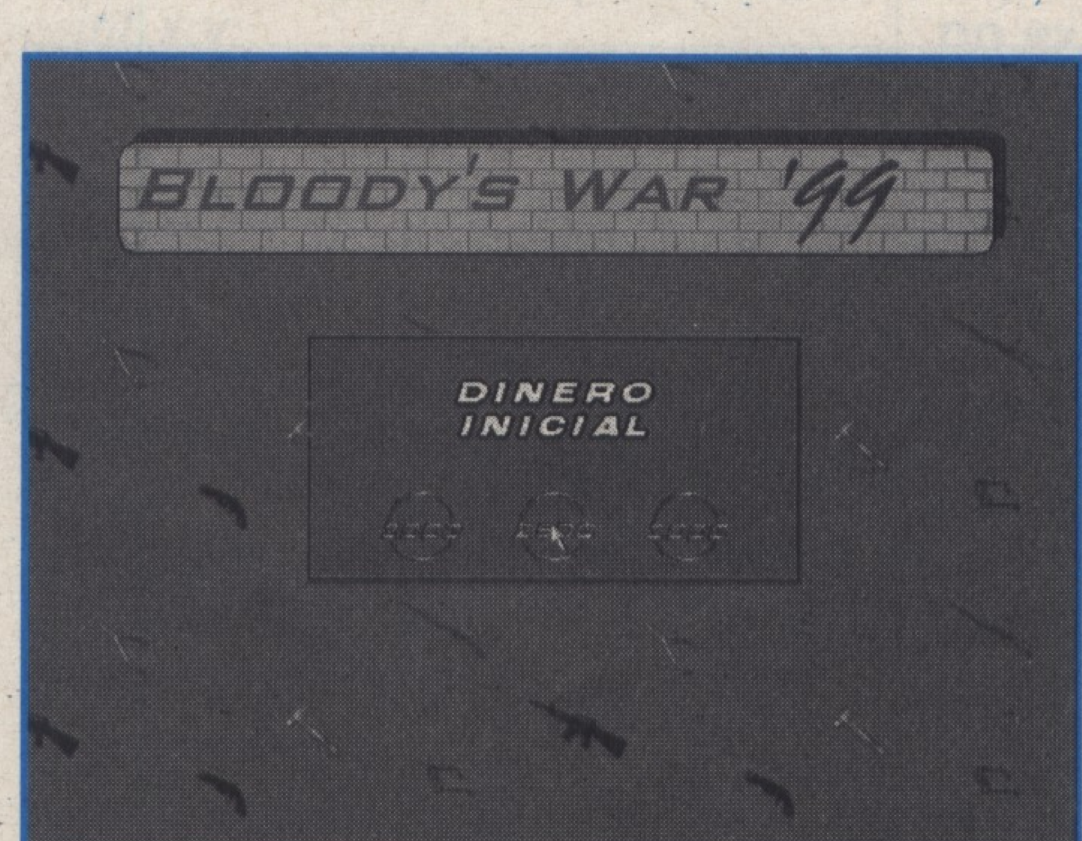
vel=2;
graph=graph1;
```

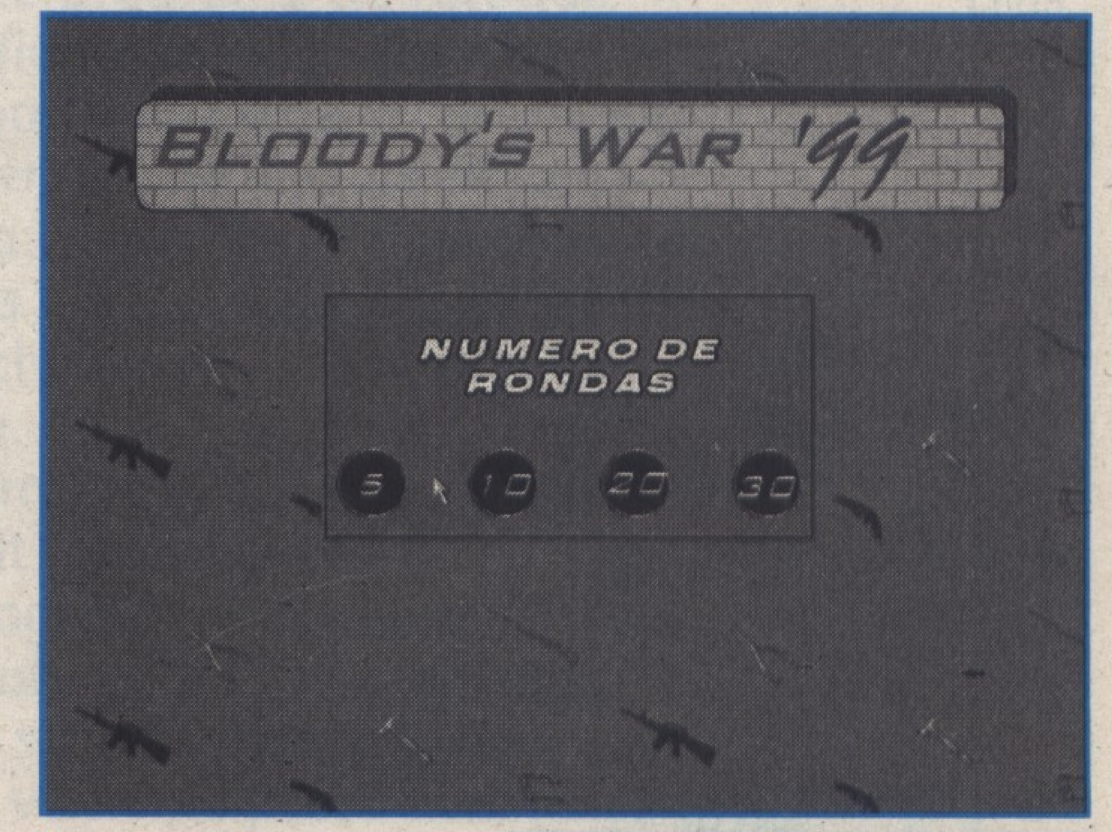


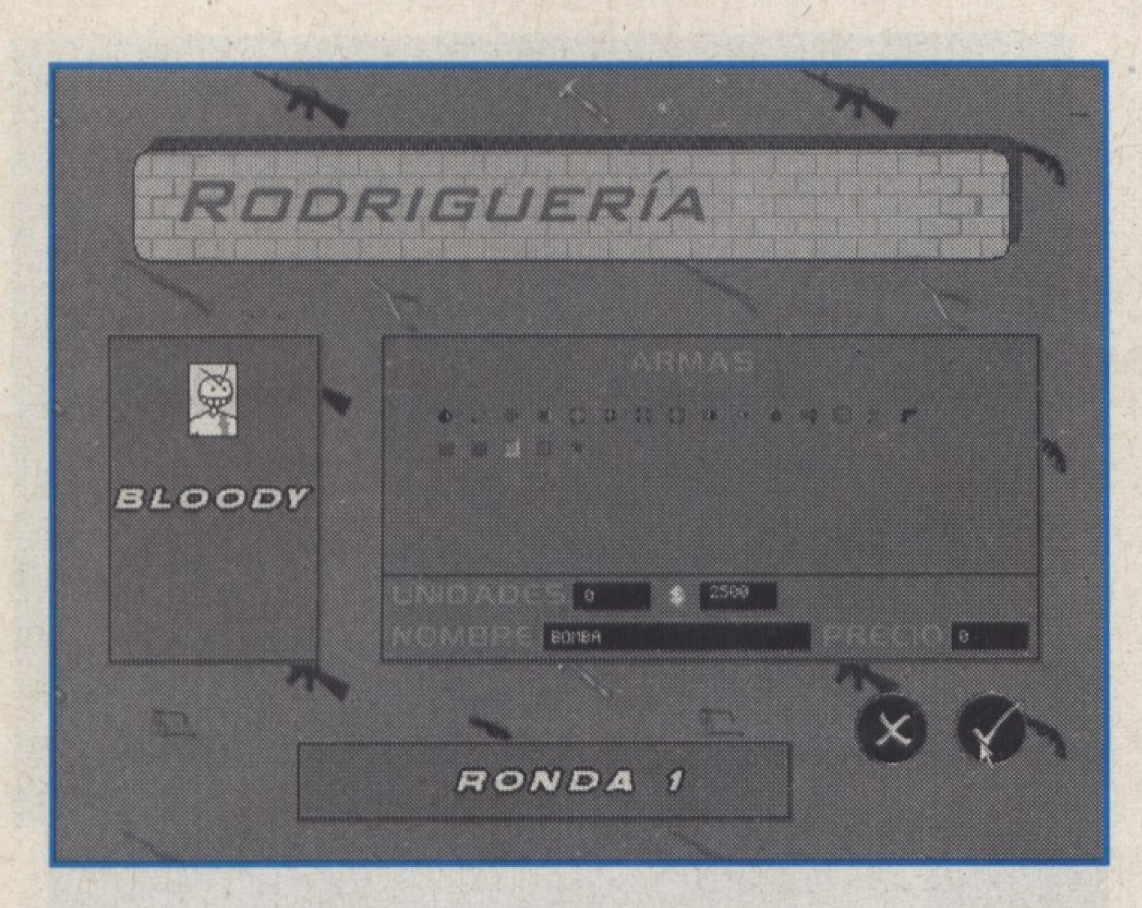

### Tabla de valores en el mapa

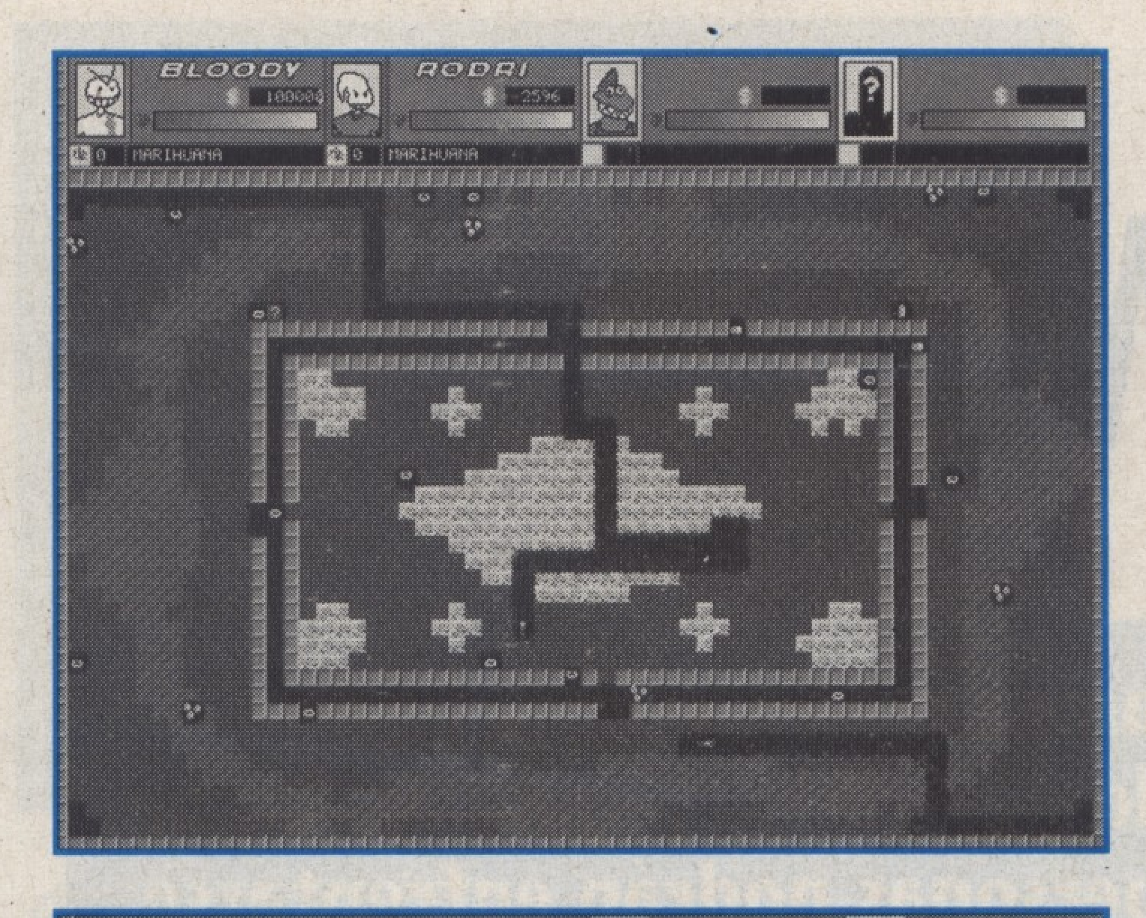
0 = roca cavada
1 = roca blanda
2 = roca semiblanda
3 = roca semidura
4 = roca dura
100 = bloque de hierro (indestructible)
101 = oro
200 = hay algún objeto o arma

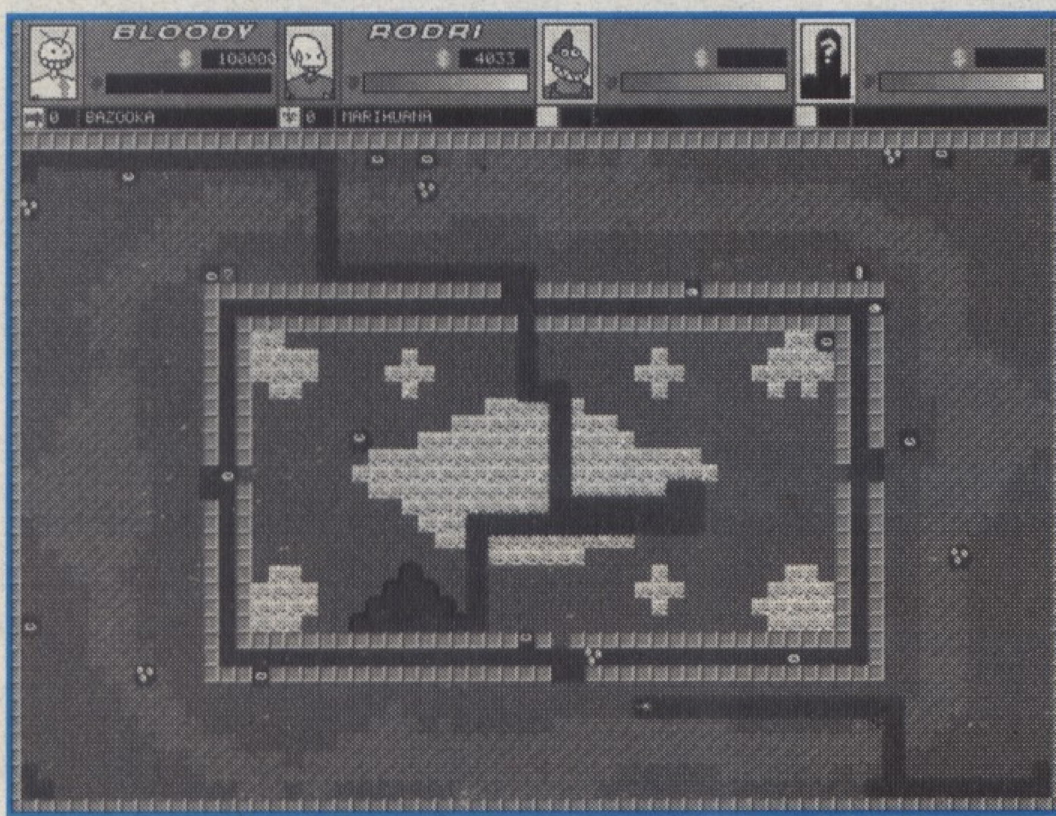
Siempre que el valor MAPA[cy*mWIDTH+cx] de un player no sea 0, se llama al proceso cavar.

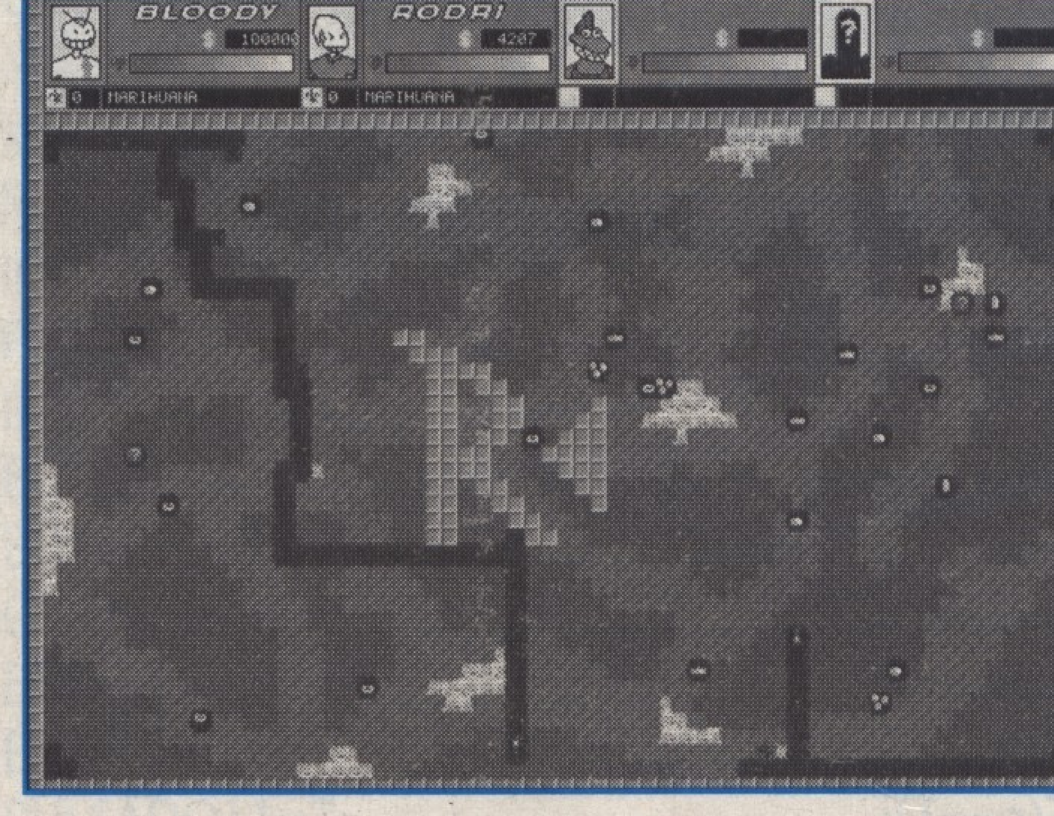





```
MAPA[cy*mWIDTH+cx]=0;
put(f_juego,49,cx*10+5,cy*10+5);
LOOP

x=cx*10+5;
y=cy*10+5;

IF(key(P_up[P]) AND NOT key(_right) AND
NOT key(_left))
  graph=graph1;
  scan[P]=P_up[P];
  IF(MAPA[(cy-1)*mWIDTH+cx]==0)
  var1=0;
  LOOP

IF(key(P_put[P]))activar_arma(cx,cy,P);END
  y-=vel;
  var1++;
  IF(var1==10/vel)cy--;BREAK;END
   FRAME;
  END
  ELSE
   chispa(x,y-10,rand(-1,1),rand(1,2),P);
   cavando=cavar(P_up[P],3,7,cx,cy-1,p);
  END

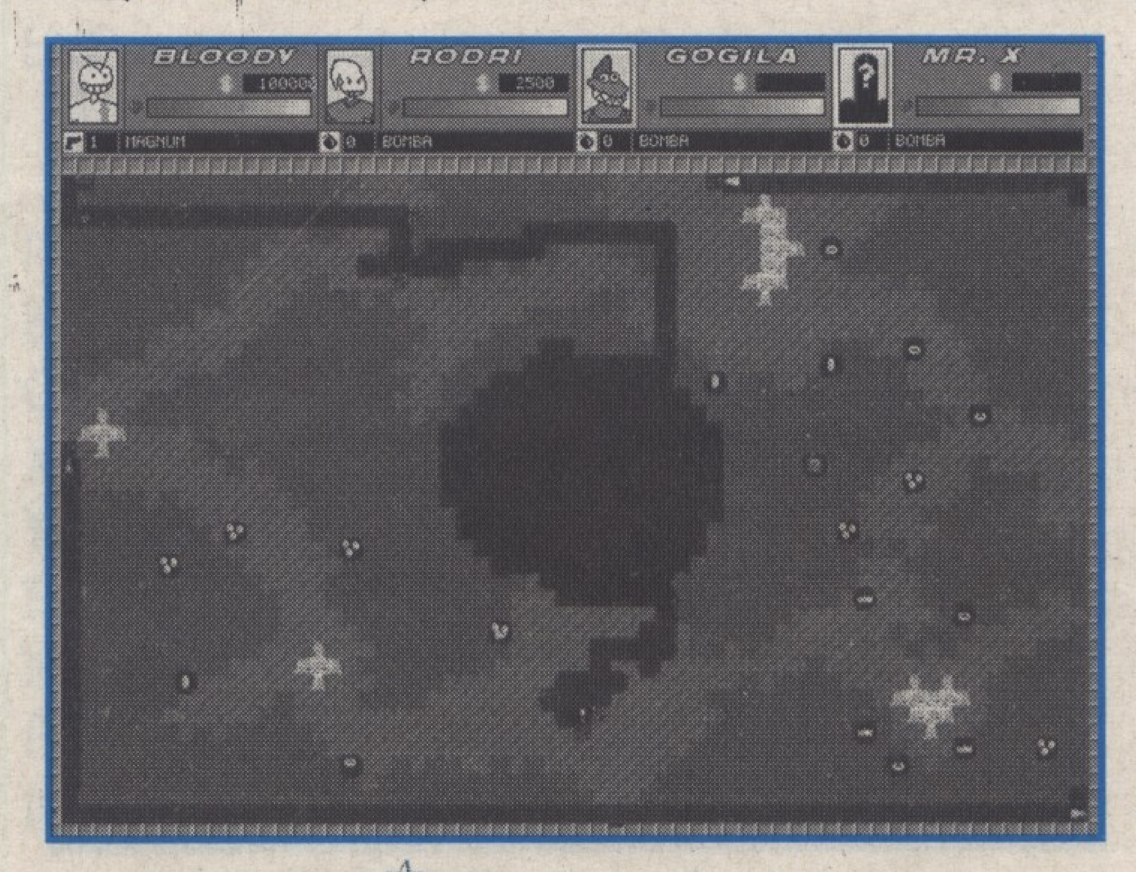
END

IF(key(P_down[P]) AND NOT key(_right) AND
NOT key(_left))
  graph=graph2;
  scan[P]=P_down[P];
  IF(MAPA[(cy+1)*mWIDTH+cx]==0)
  var1=0;
  LOOP

IF(key(P_put[P]))activar_arma(cx,cy,P);END
  y+=vel;
  var1++;
  IF(var1==10/vel)cy++;BREAK;END
   FRAME;
  END
  ELSE
   chispa(x,y+10,rand(-1,1),rand(-2,-1),P);
   cavando=cavar(P_down[P],4,8,cx,cy+1,p);
  END
END

IF(key(P_right[P])AND NOT key(_up) AND
NOT key(_down))
  graph=graph3;
  scan[P]=P_right[P];
  IF(MAPA[cy*mWIDTH+cx+1]==0)
  var1=0;
  LOOP

IF(key(P_put[P]))activar_arma(cx,cy,P);END
  x+=vel;
  var1++;
  IF(var1==10/vel)cx++;BREAK;END
```




```
  FRAME;
  END
  ELSE
   chispa(x+10,y,rand(-2,-1),rand(-1,1),P);
   cavando=cavar(P_right[P],1,5,cx+1,cy,p);
  END
END

IF(key(P_left[P])AND NOT key(_up) AND N
key(_down))
  graph=graph4;
  scan[P]=P_left[P];
  IF(MAPA[cy*mWIDTH+cx-1]==0)
  var1=0;
  LOOP

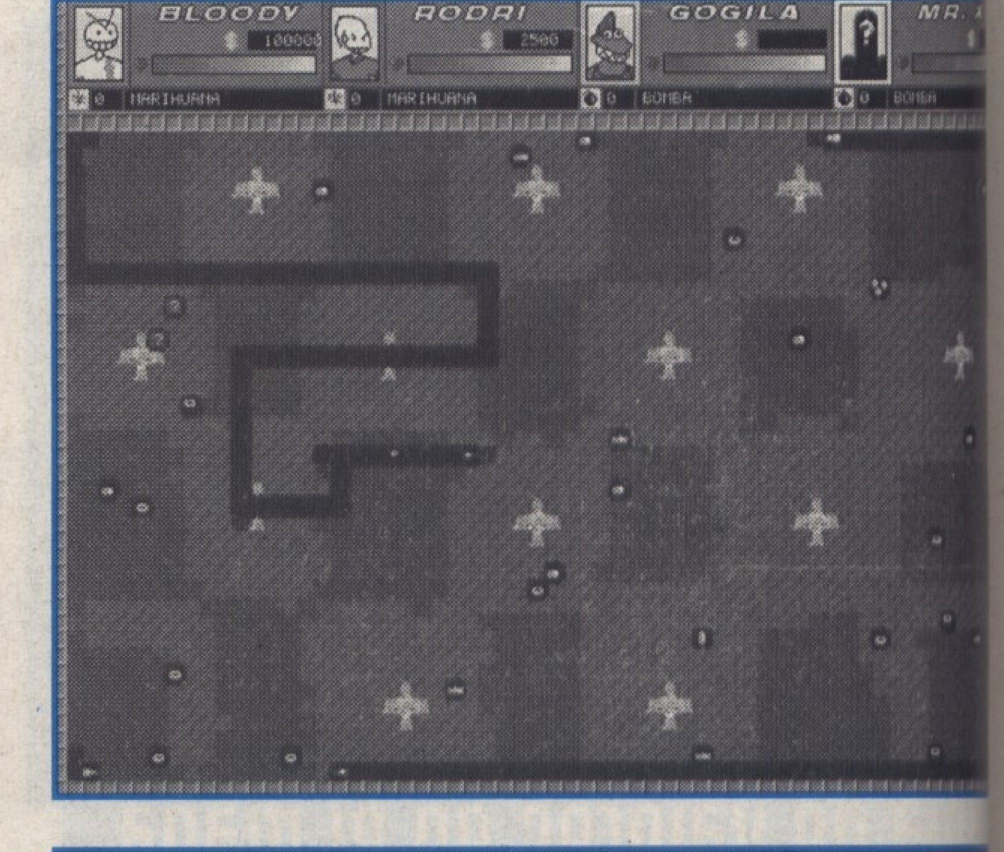
IF(key(P_put[P]))activar_arma(cx,cy,P);END
  x-=vel;
  var1++;
  IF(var1==10/vel)cx--;BREAK;END
  FRAME;
  END
  ELSE
   chispa(x-10,y,rand(1,2),rand(-1,1),P);
   cavando=cavar(P_left[P],2,6,cx-1,cy,p);
  END
END

IF(key(P_put[P]))activar_arma(cx,cy,P);END

IF(key(_esc))FIN=2;END

FRAME;

END
END
```






pularlo de forma directa antes de volcarlo al *buffer* posterior. Se puede acceder a dicho puntero mediante: *char *background.*

DIV nos permite, además de manipular los datos de estas imágenes, realizar diversas opciones de manipulado entre los distintos pasos del proceso. Principalmente utilizaremos las funciones:

- *void post_process_buffer(void)*: permite realizar modificaciones en el *buffer* posterior una vez dibujados todos procesos y el fondo, y antes de dibujarlo en la pantalla.
- *void background_to_buffer(void)*: realiza la copia del fondo de pantalla al *buffer* posterior.

Además de éstos, existen otras muchas funciones que iremos viendo poco a poco según las necesitemos.

Para que estas funciones hagan efecto en nuestras Dlls, debemos indicarle a DIV que vamos a usarlas al igual que hicimos con el resto de funciones diseñadas anteriormente. Pero al estar ya declaradas, debemos indicárselo de forma distinta. Para ello en vez de recurrir a la función COM_export() utilizada para el resto de funciones, utilizaremos la función DIV_export() que tiene el siguiente prototipo:

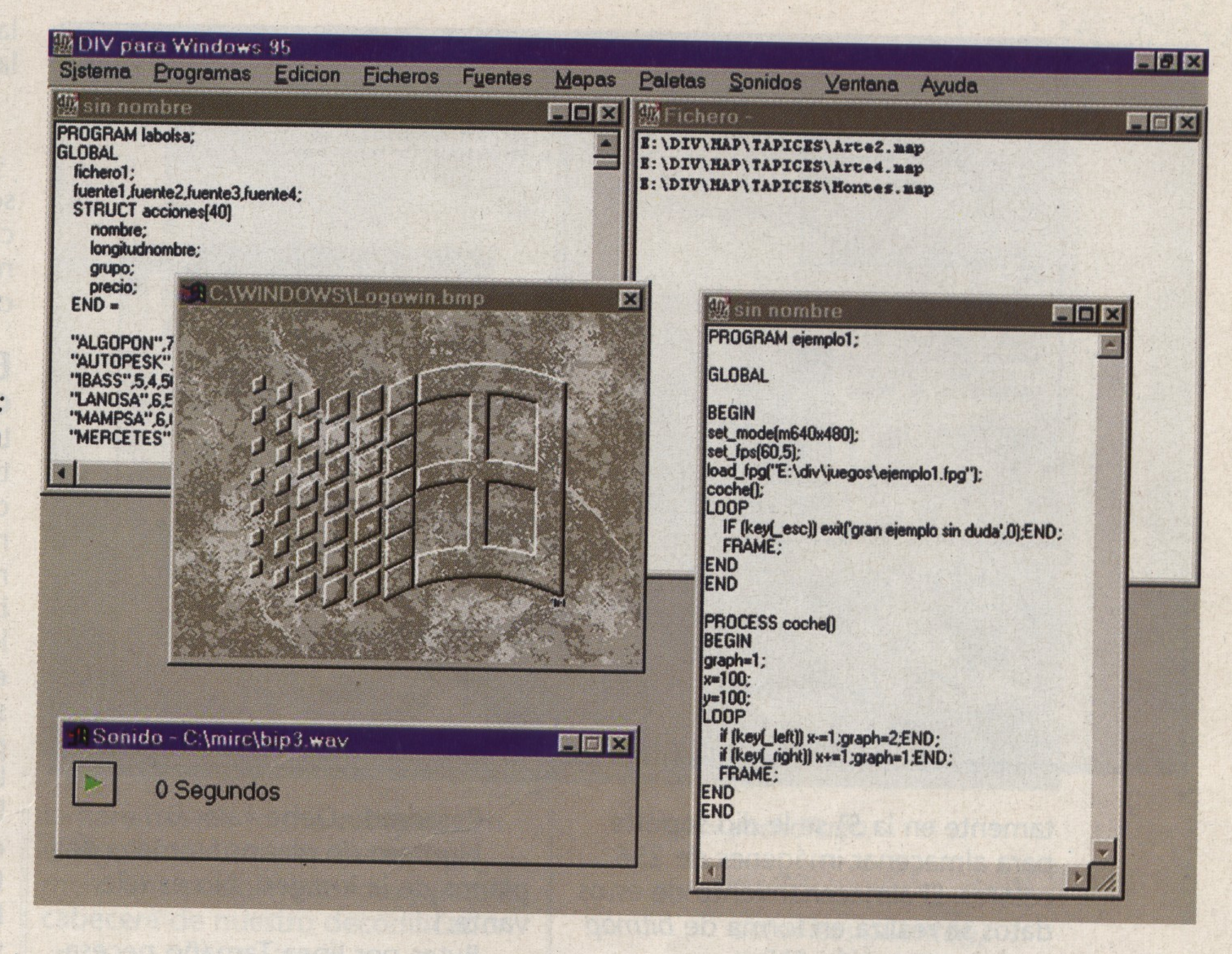
```
DIV_export(Apost_process_buffer@,post_process_buffer);
DIV_export(Abackground_to_buffer, background_to_buffer);
```

Estas funciones deben ejecutarse dentro de la función *divmain()*, de forma que si utilizamos una de las funciones anteriores, debemos comunicárselo a DIV de la siguiente forma:

```
void __export divmain(COMMON_PARAMS) {
    GLOBAL_IMPORT();

DIV_export(Apost_process_buffer@,post_process_buffer);
}
```

Tomemos un ejemplo de lo anterior para dibujar, por ejemplo, un recuadro con el color 50 en la esquina inferior derecha de tamaño 100x50 en todos los frames. Para ello, debemos alterar el contenido del *buffer* posterior antes de volcarlo a la pantalla. Para hacerlo recurriremos a la función *post_process_buffer()*. Pero debemos también conocer las dimensiones de la pantalla para colocar dicho recuadro en




esa posición. Para ello disponemos de las siguientes variables:

## Variable Significado

*wide* Ancho de la pantalla en puntos (x)
*height* Alto de la pantalla en puntos (y)

Hemos de aclarar que además de poder utilizar como puntero la variable de acceso al buffer posterior (buffer), podemos usarla como una matriz de tamaño *wide*height*. Veamos cómo se realizaría con ambos sistemas:

```
void post_process_buffer(void)
{
 int x,y;

 for (y = height-50;y<heigth;y++)
   for (x = wide-100;x<wide;x++)
     buffer[wide*y+x]=50;
 }  void post_process_buffer (void)
 {
  int x,y;
  char *puntero;

  puntero=buffer;

  puntero+=wide*(height-50)+(wide-100);
  for (y=1;y<=50;y++)
  {
   for (x=1;x<=100;x++)
     (* puntero)++ = 50;
   puntero+=wide-100;
  }
 }
```

Ejemplo de uso de *buffer* como puntero o como matriz en el relleno de una zona cuadrada. Se observa la mayor sencillez del uso de matrices en este caso.

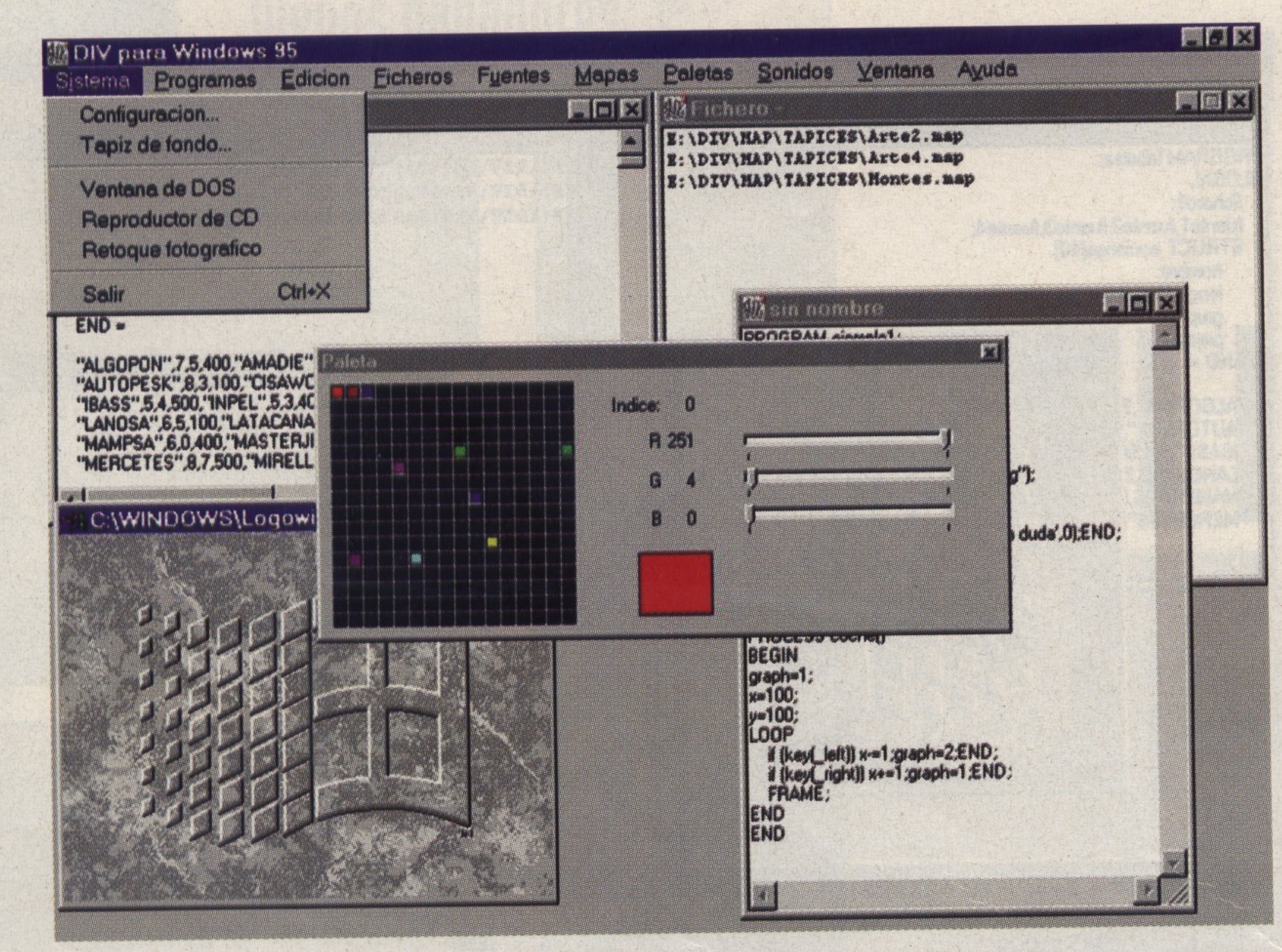
La sencillez de cada uno de los métodos depende del caso. Si vamos a rellenar una pantalla por completo, el uso de punteros es más claro y rápido. Sin embargo, para rellenar zonas cuadradas en distintos puntos de la pantalla es mucho más engorroso, como podemos ver en el cuadro 1.

Una vez visto todo este proceso, vamos a realizar una función para cargar archivos PCX en el fondo de la pantalla. Pero para ello debemos ver cuál es la estructura de este tipo de ficheros.

## Estructura de los ficheros gráficos PCX

Este tipo de ficheros tal vez sea uno de los más sencillos de utilizar y de una gran difusión. Podríamos decir que actualmente no existe programa de retoque fotográfico que no lo soporte. DIV soporta su carga y conversión a MAP en el entorno de desarrollo. Pero queremos ir más allá y veremos cómo realizar la carga de estos ficheros en el fondo de la pantalla.

El formato PCX fue desarrollado por Zsoft Corporation para su uso en el programa de retoque Paintbrush. En sus orígenes tenía tan solo soporte para 2 colores, pero en versiones posteriores (exac-




tamente en la 5) se le dio soporte para almacenar imágenes de 256 colores. El almacenamiento de estos datos se realiza en forma de *bitmap* y con compresión RLE.

Como muchos archivos gráficos, está formado por una estructura o cabecera que nos facilita todos los datos que necesitemos acerca del tamaño de la imagen, colores y demás, una sección donde se encuentra la imagen codificada y una paleta de colores. Analicemos en primer lugar la cabecera de estos archivos:

## La cabecera de las imágenes PCX

Tipo:Descripción:Tamaño (en bytes).

Distintivo:Valor constante que indica que se trata de un archivo PCX. Su valor es 0x0A (10 en decimal):1.

Versión:Versión del archivo. En nuestro caso debe ser la versión 5 (valor 5):1.

Codificación:Si su valor es 1, tiene codificación RLE:1.

Bits por pixel (bpp):256 colores son 8 bits por pixel (valor 8):1.

Xmin:Coordenada x de la esquina superior izquierda:2.

Ymin:Coordenada y de la esquina superior izquierda:2.

Xmax:Coordenada x de la esquina inferior derecha:2.

Ymax:Coordenada y de la esquina inferior derecha:2.

Vres:Resolución en ppp (puntos por pulgada) vertical. No nos importa:2.

Hres:Resolución en ppp (puntos por pulgada) vertical. No nos importa:2.

Paleta 16 colores:Valores RGB para archivos de 16 colores. No nos interesa:48.

Reservado:Debe valer 0:1.

Número de planos:Número de planos de la imagen. No es relevante:1.

Bytes por línea:Tamaño necesario de un buffer de decodificación de líneas:2.

Tipo de paleta:1 si usamos color o blanco y negro y 2 si usamos escala de grises:1.

Xtamaño:Dimensión X de la pantalla en pixels:2.

Ytamaño:Dimensión Y de la pantalla en pixels:2.

Reservado:Todos a 0 (para que la imagen comience en el byte 128):54.

Con todos estos datos disponemos ya de toda la información necesaria para conocer las características de la imagen. Hemos de notar que para saber el tamaño de nuestra pantalla debemos recurrir a los valores máximos y mínimos de las coordenadas de la pantalla de la siguiente forma:

Ancho = Xmax-Xmin+1;

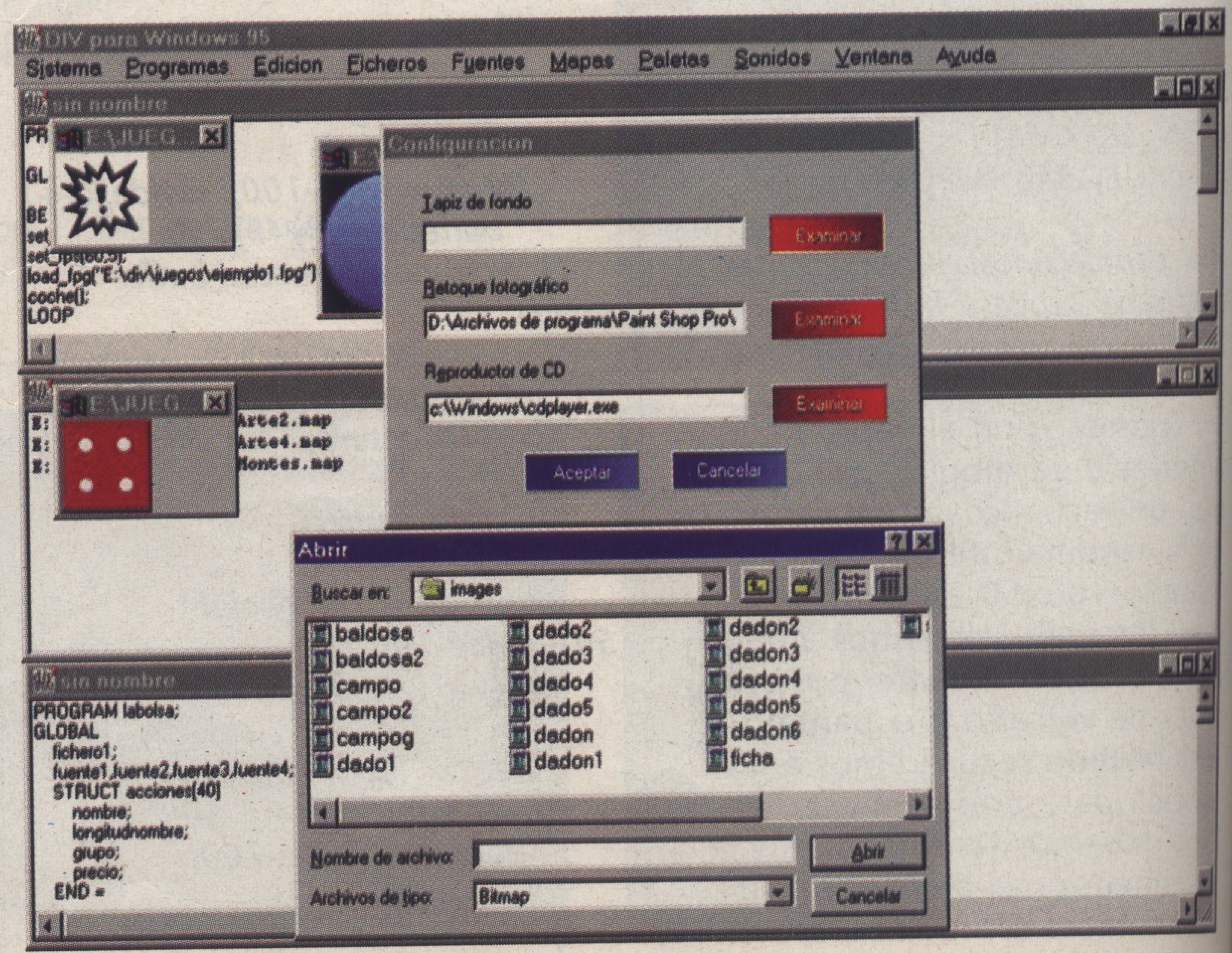
Alto=Ymax-Ymin+1;

El resto de datos tan solo debe servirnos para verificar que las características de nuestras imágenes sean las correctas. Pasemos a conocer el formato de la imagen.

## El formato de compresión RLE

La compresión RLE tiene un formato sencillo y, en el caso de 256 colores, bastante efectivo. Los colores a mostrar se representan mediante números de 8 bits (1 byte). Si no existiera compresión, los puntos se irían almacenando en el fichero de uno en uno indicando su índice correspondiente en la paleta, recorriendo horizontalmente la imagen. Pero entonces la pantalla ocuparía mucho (el tamaño sería como el de un archivo BMP, que no tiene compresión). Por tanto lo que buscamos es una repetición de varios puntos seguidos con un mismo color dentro de la imagen. El problema radica en cómo podemos indicar que un color debemos repetirlo. Esto se indica mediante la puesta a 1 de los dos bits más significativos de cada color, seguido de 6 bits que indican el número de veces que se repite el color cuyo índice se encuentra en el siguiente byte. Veamos una secuencia de bytes como ejemplo:

| Secuencia (8 bits) | Repeticiones | Color |
|---|---|---|
| 10000100 | 132 | 1 |
| 00100001 | 33 | 1 |
| 01000110 | 70 | 1 |
| 11001011 | - | 11 |
| 00001001 | 5 | - |



En los tres primeros casos los 2 bits más significativos no estaban al mismo tiempo a 1, con lo que representaban al color de un solo pixel. En el cuarto, sí tenemos los 2 bits más significativos a 1 con lo que repetimos el siguiente byte 001011 veces (11 en decimal), cuyo valor es 5.

Es importante notar tres conceptos que suelen confundirse:

- Si encontramos un byte de repetición con los dos bits más significativos a 1, si el siguiente byte tiene también ambos bits a 1, no significa repetición, sino que representa a un color. Por tanto si sustituimos al quinto elemento por 11000010 significará repite 11 veces el color 194.
- Muchos se habrán preguntado qué pasa si queremos representar un número mayor que 192 (11000000) en RLE. Pues bien, se hará de forma que digamos lo siguiente: repite una vez, el color C. Por tanto si queremos representar el número 200 una vez tendremos la secuencia: 11000001,11000100.
- Como consecuencia del primer punto y por la limitación de 6 bits de repetición, el número máximo de repeticiones será de 63 veces (111111). La repetición no es acumulativa como dijimos antes.

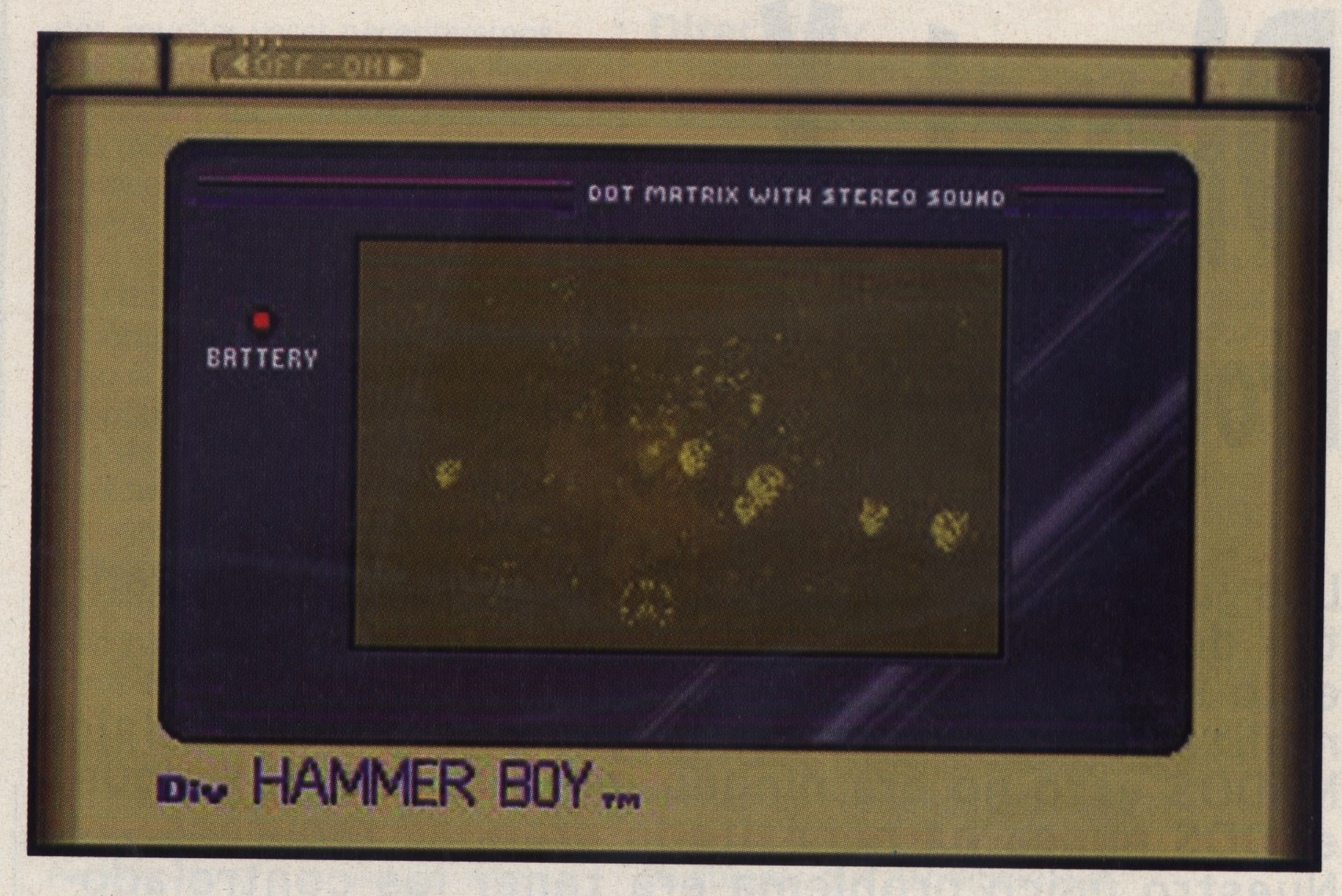
Con esto tenemos ya todos los conocimientos para decodificar un código RLE. Tan solo necesitamos conocer la estructura de la paleta para comenzar ya a trabajar.

## La paleta del PCX

Una vez alcanzado el último punto codificado de la imagen, comienzan los datos de la paleta. Son 768 bytes donde encontramos 256 tripletas de bytes, una para cada color y con sus valores RGB. Debemos recordar que el uso de paletas en DIV está restringido a valores de 0 a 63 en los datos de color RGB, con lo que debemos dividir por 4 el valor de estos datos de paleta (o realizar un desplazamiento a la derecha de 2 bits que es mucho más rápido) para enviar estos datos a DIV. Por ahora no tocaremos estos datos, ya que lo trataremos con detenimiento en el siguiente número.

## Algoritmo de descompresión

Vamos a ver de forma práctica cuál sería una posible estructura del programa descompresor de PCX. No vamos a darle aún estructura de dll para no aumentar la complejidad en su compresión, y porque lo haremos de forma completa en el siguiente número. Veamos primero una posible estructura para la cabecera de nuestro decodificador:




```
typedef byte unsigned char;
typedef word unsigned char;

struct cabecera
{
    byte marca;
    byte version;
    byte codificacion;
    word xmin,ymin,xmax,ymax;
    word hres,vres;
    byte paleta16[48];
    byte reservado;
    byte planos;
    word bpl;
    byte tipopal;
    word xtam,ytam;
    byte reserva[54];
}
```

Esta estructura se irá rellenando con los datos del fichero, pero tan solo nos interesarán los datos de alto y ancho para conocer las dimensiones de la imagen y comprobar la versión, marca y codificación por simple seguridad. El algoritmo de decodificación de forma esquemática sería el del cuadro 1.

```
void decodifica (void)
{
    // consideremos lee_siguiente()
como la función que devuelve el
siguiente byte del fichero PCX
    // y *destino como el puntero
a memoria o a la pantalla
    int max,contador,ancho,alto;
    byte dato,repeticion,i;
    ancho=cabecera.xmax-cabece-
ra.xmin+1;
    alto=cabecera.ymax-cabece-
ra.ymin+1;
    max=ancho*alto;

    do{
        dato=leesiguiente();
        if (dato && 192)
        {
         (*destino)++=dato;
        contador++;
        }
        else
        {
        repeticion=dato-192;
        dato=leesiguiente();
        for (i=0;i<repeticion;i++)
(*destino)++=dato;
        contador+=repeticion;
        }
    }
    while (contador<max);
}
```

Cuadro 1. Algoritmo de descompresión PCX.

Con esto tenemos ya casi todo lo que necesitamos para desarrollar esta nueva función visual. En el próximo número hablaremos de las estructuras para alterar las paletas de DIV y el acceso a archivo mediante dlls, para, de esta forma, tener ya todas las herramientas necesarias para terminar con el desarrollo de estas funciones de decodificación y muestreo de archivos PCX.

Si alguno de los lectores se anima a enviar sugerencias, mejoras al algoritmo, o va más allá y acaba con la librería de PCX, pueden enviarlo a la dirección trinidad@arrakis.es y serán comentadas. Suerte y buena decodificación.

**Pablo Trinidad**

# Direct X

## Programación avanzada

**Los juegos siempre han tenido unos requerimientos de proceso y de recursos bastante altos; esto se debe principalmente a la necesidad de realizar operaciones complejas con gráficos. Bajo MSDOS esto no era ningún inconveniente, ya que el DOS nunca ha necesitado grandes recursos. Nuestro único problema era tener los controladores adecuados para acceder a la memoria que necesitáramos.**

La gran velocidad de la que siempre han hecho gala los juegos para DOS, tenía su explicación en el acceso directo a los dispositivos. La programación resultaba muy complicada, pero tenía su recompensa. Sin embargo, el principal inconveniente de la programación bajo DOS, era la diversidad de dispositivos.

La presencia de tarjetas de vídeo de diferentes casas, de diferentes modos de vídeo, así como de la falta de acuerdo entre los fabricantes de tarjetas para asumir un consenso con las normas SVGA, hacían de la programación de juegos una verdadera locura. Se programaba para modos concretos y tarjetas especificas de vídeo, por lo que era normal que juegos muy avanzados solo funcionaran con tarjetas de vídeo concretas.

### Windows sobre el DOS

La aparición de Windows 3.1 provoco una revolución en el mundo del PC. Hacía tiempo que los Macintosh disponían de una interfaz gráfica, fácil de usar, cómoda y relativamente potente. La respuesta de Microsoft fue Windows, que tras varias versiones de dudosa calidad, se estabilizo en la versión 3.1.

La programación de juegos bajo DOS supone conocer muy bien los dispositivos físicos

Windows tenía una ventaja sobre el DOS en el que se apoyaba: se insertaba una capa de software entre el hardware y los programas, de tal forma que el programador no tenia necesidad de preocuparse de la tarjeta de vídeo ni de recursos específicos. Todo esto, a pesar de ser una innegable ventaja, tiene un gran inconveniente: es bastante lento, así como un gran devorador de recursos.

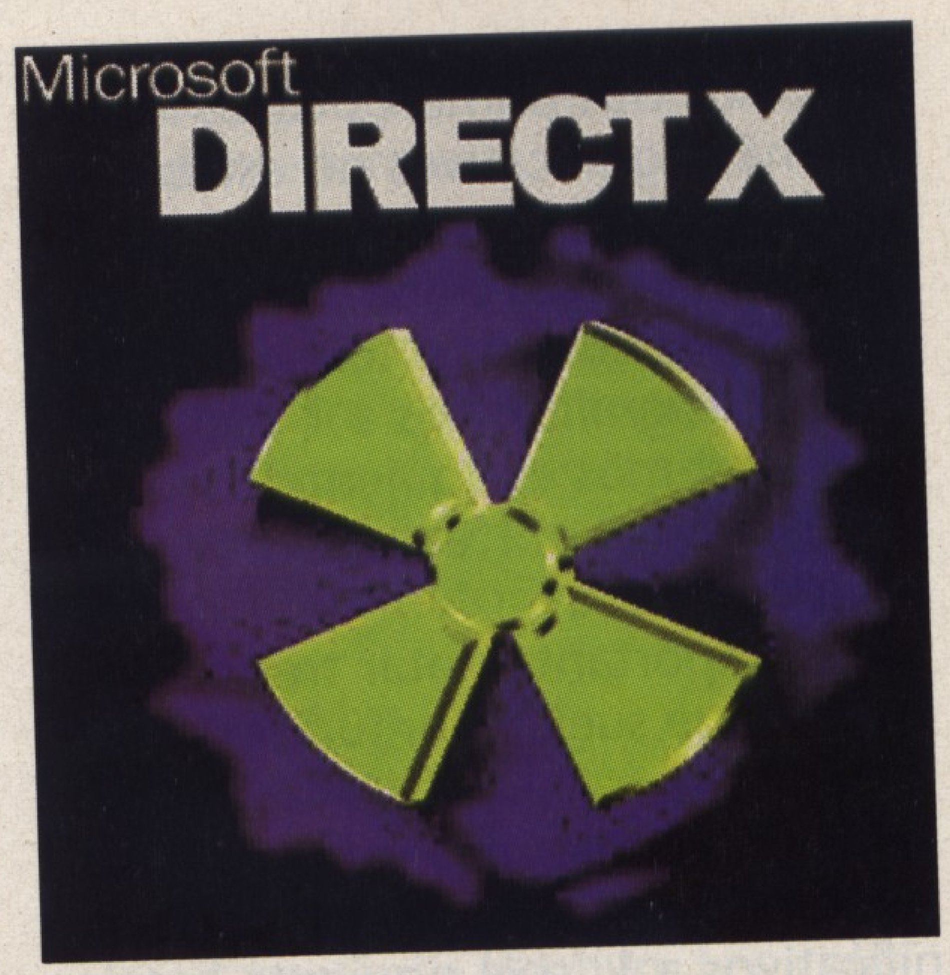
Lejos de poder aprovechar las nuevas posibilidades de Windows, el programador de juegos, se encontraba con la imposibilidad de acceder directamente a los recursos. Esto hacia inviable conseguir juegos potentes, limitándose los pocos juegos existentes a pequeñas animaciones con gráficos pobres.

### Windows 95

A pesar de Windows 3.1 y de su versión para red 3.11, el uso de un PC no era en absoluto sencillo. Windows 95 nació en respuesta a las necesidades de usuarios que veían en el DOS y en la configuración del sistema una tarea demasiada compleja.

A pesar de toda la expectativa levantada por Windows 95, en lugar de tener un sistema operativo nos encontramos con un nuevo Windows 3.1, con mayores requerimientos y nuevas dificultades para el programador. De modo que los juegos siguieron diseñándose para el DOS, obligando en muchos casos a prescindir de Windows.

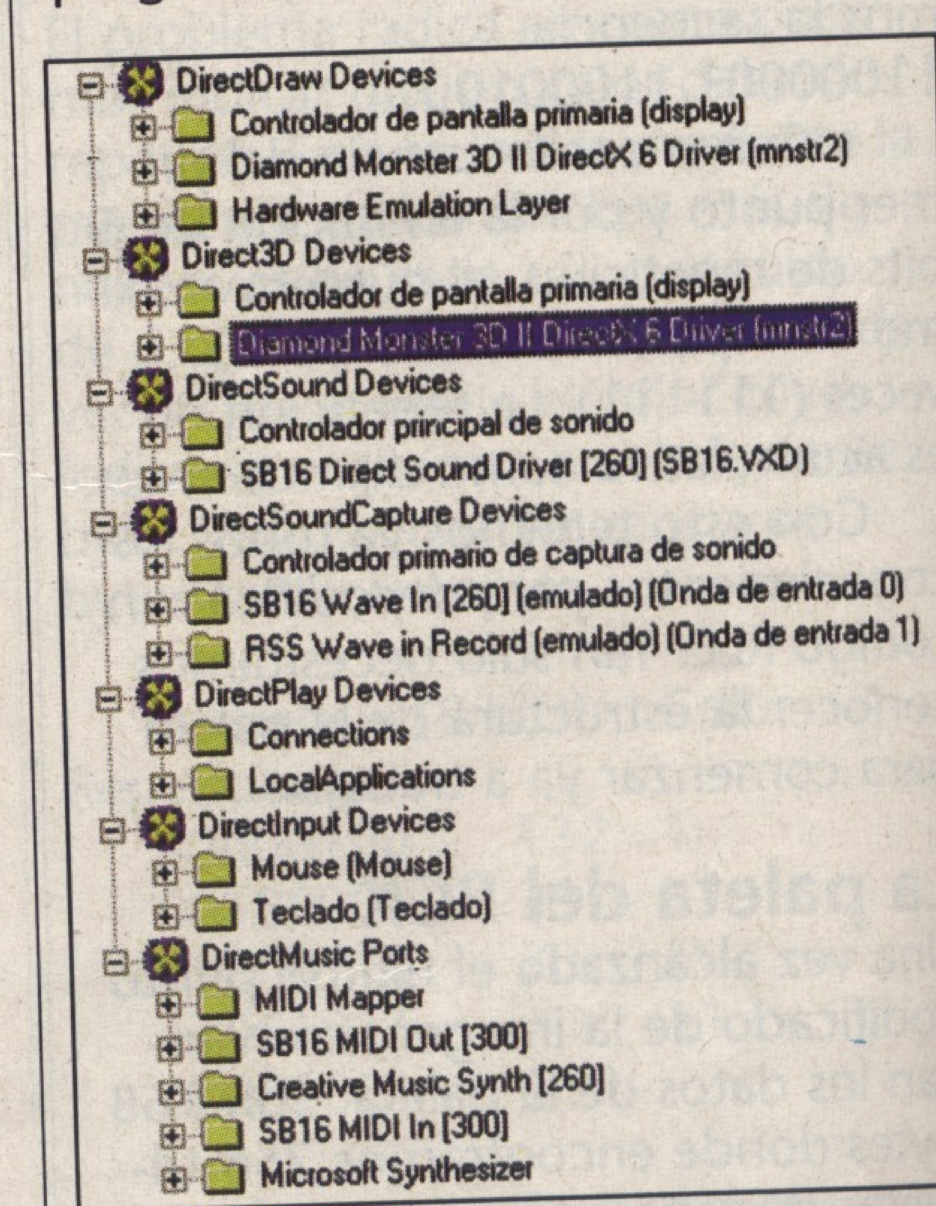
Microsoft no tardo en darse cuenta de que si quería hacer de Windows 95 un verdadero triunfo los juegos debían funcionar en Windows; y para eso debían dar al programador una razón para cambiar de forma de pensar.




**DirectDraw nos permite programar juegos rápidos gracias al HAL que DirectX inserta entre nuestra aplicación y el hardware.**

### DirectX

DirectX es la respuesta de Microsoft al programador de videojuegos. Un conjunto de funciones, agrupadas en librerías, creadas en respuesta a la carencia de Windows para soportar programación avanzada, que permitiera a los programadores aprovechar toda la




**DirectX es, básicamente, un conjunto de funciones agrupadas en librerías tal como se ve en la imagen.**

potencia que el procesador y la tarjeta gráfica pueden ofrecer.

Si quisiéramos dar una definición de lo que son las DirectX, podríamos decir que se trata de un conjunto de rutinas destinadas a acceder directamente al hardware del sistema de forma rápida y eficaz. En otras palabras, son un boquete en Windows que nos permite tocar los dispositivos físicos, evitando los obstáculos interpuestos hasta el momento.

La principal virtud de DirectX reside en su capacidad para hacernos olvidar qué dispositivo posee el usuario. Nos es indiferente si se tiene una RivaTNT o una 3dfx, una SoundBlaster o una Yamaha.



**DirectX permite que aplicaciones con altos requerimientos funcionen eficazmente.**

DirectX incorpora un *driver* específico para cada dispositivo; cada fabricante desarrolla el software necesario para poder acceder directamente al dispositivo, de esta forma, podemos utilizar las mismas funciones sin necesidad de preocuparnos por las particularidades de cada dispositivo.

Por si fuera poco, si un dispositivo requerido por nuestro programa no se encuentra presente, DirectX intenta emularlo por software, aunque no con un gran resultado como es lógico. Esta habilidad de DirectX de emular un hardware inexistente (HEL), evita al programador y al usuario muchos quebraderos de cabeza, asegurando que, en la mayoría de los casos, nuestro juego va a poder ejecutarse.

Desde su concepción, el conjunto de rutinas ha pasado a ser una herramienta bastante cumplidora. Poco a poco, Microsoft va añadiendo diferentes funciones para aplacar las ansias de los programadores, cubriendo distintos aspectos: gráficos en dos y tres dimensiones, sonido, dispositivos de control, comunicación de ordenadores y música. Actualmente las DirectX se encuentran en su versión 6.1, y Microsoft camina hacia la versión 7.0, que integrará, si los planes no cambian, con Windows 2000.

Cada uno de los diferentes aspectos de la programación de videojuegos queda cubierta con las distintas librerías que Microsoft ha preparado:

**DirectDraw:** rutinas de dibujo en dos dimensiones.
**DirectInput:** dispositivos de entrada.
**DirectSound:** sonidos.
**DirectMusic:** música.
**Direct3D:** gráficos en tres dimensiones.
**DirectPlay:** comunicación de ordenadores.
**DirectSetup:** instalación automática de las DirectX.

A pesar de ser objetos separados, muchas veces necesitaremos realizar operaciones con una librería diferente para continuar con nuestras tareas. Por ello, necesitaremos conocer, si no todos los aspectos, al menos las principales funciones.

## DirectDraw

El conjunto de rutinas agrupadas bajo el nombre de DirectDraw, nos proporciona acceso directo a la tarjeta de vídeo, posibilitándonos realizar operaciones relacionadas con gráficos bidimensionales con una gran facilidad y velocidad. Podemos preparar nuestra aplicación para trabajar tanto a pantalla completa como en modo ventana.

## DirectInput

Los juegos necesitan algo más que un teclado o un ratón. El joystick nació como una emulación de los mandos de las recreativas, llegando en al actualidad a superarlos en algunos casos. Responden a la fuerza con que lo manipulemos, modificando su tacto según el terreno, vibran, se quedan bloqueados en una posición, etc. DirectInput nos facilita el acceso de todos los dispositivos de entrada que el usuario quiera usar, desde el teclado hasta el más avanzado joystick.

## DirectSound

Reproducir sonido en tiempo real y sincronizarlo con las animaciones que discurren a través de nuestro monitor, no es tarea sencilla. DirectSound nos posibilita todo eso y más. Reproducir y grabar sonido en tiempo real, reproducir sonidos en tres dimensiones, son algunas de las tareas que podremos realizar sin tener que realizar grandes esfuerzos gracias a DirectSound.

## DirectMusic

Reproducir música, una tarea aparentemente sencilla, no lo es tanto si tenemos en cuenta que tenemos que realizar un gran número de operaciones para mostrar nuestros gráficos. Hasta la versión seis de DirectX, no teníamos otra forma de reproducir música si no era a través de la API de Windows, ahora, podemos acceder directamente a la tarjeta de sonido.

## Direct3D

La programación de gráficos en tres dimensiones requiere de una programación muy compleja. Direct3D nos facilita el esfuerzo necesario para llevar a la pantalla objetos tridimensionales, dándonos funciones capaces de realizar la infinidad de operaciones necesarias para mover un objeto en un mundo virtual.

## DirectPlay

Si hemos jugado alguna vez en red, los juegos no volverán a ser los mismos. La posibilidad de jugar en red local o a través de Internet hace que un simple juego pueda llegar a convertirse en una forma de vida. DirectPlay se encarga de facilitarnos la comunicación entre varios ordenadores, sin importar donde se encuentren.

> Windows hace que la programación de juegos para este entorno produzca pobres resultados

## DirectSetup

Si el destinatario de nuestro juego no tiene en su ordenador DirectX, tenemos un problema. Mediante DirectSetup podemos solucionarlo, una vez que hallamos detectado la ausencia de DirectX, instalaremos las librerías que permitirán que nuestro juego funcione.

## Multimedia

DirectX no es tan sólo programación de videojuegos. También podemos utilizar las rutinas que nos ofrece para aplicaciones que necesitan de la potencia y velocidad que nos puede ofrecer. Podemos utilizar su velocidad para reproducir vídeo o música en tiempo real, evitando molestos cortes e interrupciones. Tarjetas capturadoras de televisión son un claro ejemplo de las posibilidades que DirectX puede proporcionar.

## Visual C++

Aunque podemos utilizar diferentes lenguajes y compiladores para programar con DirectX, no cabe duda del propósito original de Microsoft. El futuro programador debía dirigir la orquesta desde Visual C++ 5.0. El compilador de Microsoft, sin duda potente, es el más adecuado para trabajar con el SDK de las DirectX.

> Directx nos proporciona velocidad y potencia, algo que necesitan tanto los juegos como las aplicaciones multimedia

Podemos utilizar las rutinas de DirectX desde Delphi, en un futuro incluso desde Visual Basic, pero el C++ es el camino hacia una aplicación potente y rápida, todo aquello que necesita DirectX. La elección del compilador no es tan solo una cuestión de gusto, ni siquiera de comodidad. Utilizar Microsoft VisualC++ es una necesidad si queremos asegurarnos una total trasparencia en la integración de nuestro programa con las librerías de DirectX.

**Armando Vélez**

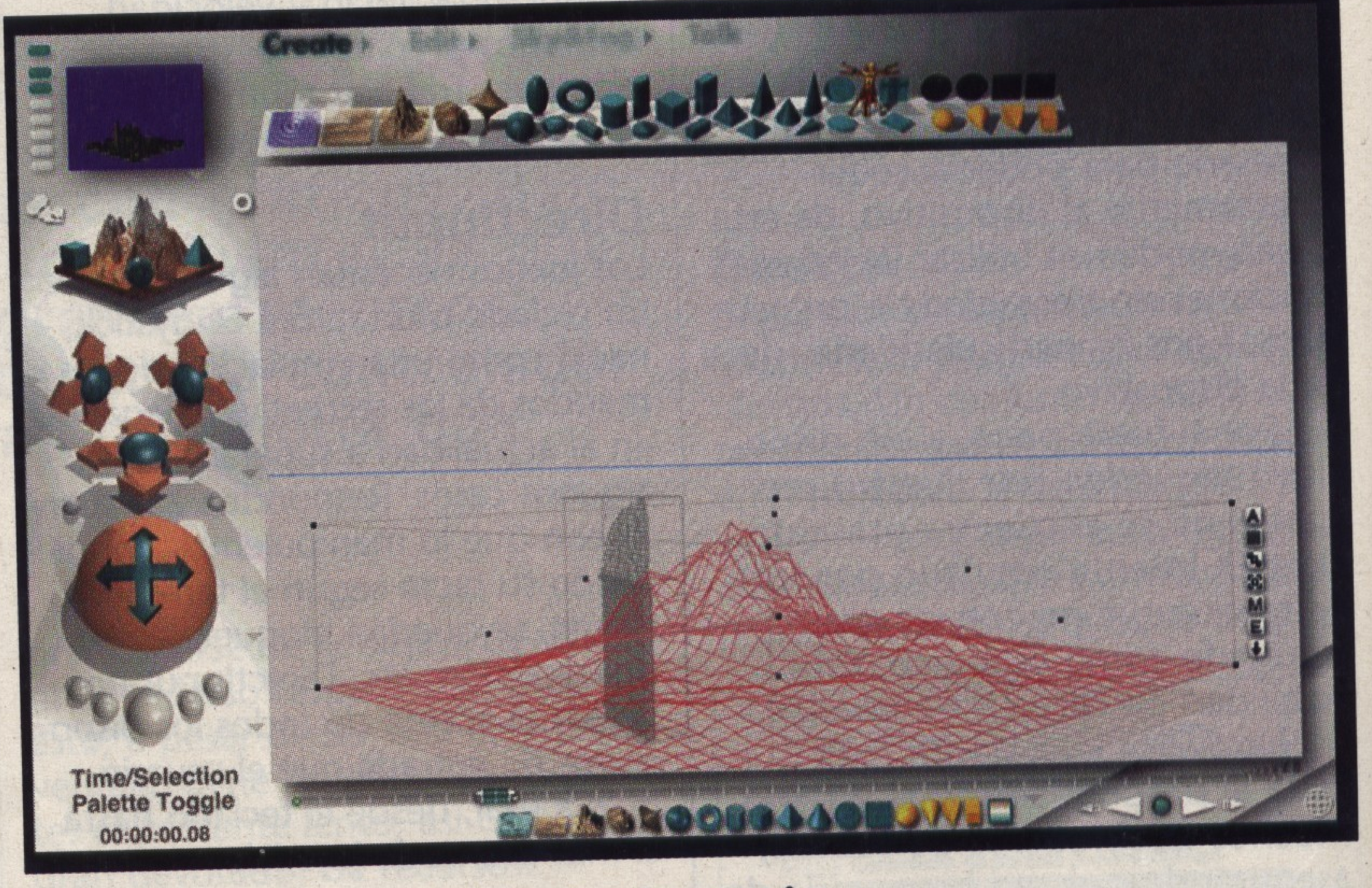
# Bryce 4

## 4 dimensiones de creatividad

**Hace unos meses 3D World regaló a sus lectores la versión completa de Bryce 2, hoy os presentamos la última y más espectacular versión de este conocido generador de entornos 3D.**

Unos meses despúes de la misteriosa desaparición de Kai Krause de MetaCreations os comentamos la última versión de uno de sus hijos más conocidos: Bryce. Bajo este nombre se esconde uno de los más potentes generadores de entornos tridimensionales del mercado. Hoy en día, cualquier infografista que se precie utiliza de vez en cuando esta potente herramienta a la hora de generar paisajes, dada su potencia, sencillez de manejo y, por supuesto, las amplias posibilidades que nos brinda.

Quizás la versión más extendida es la segunda, ya que MetaCreations decidió regalarla en algunas revistas, pero es esta nueva versión la que está levantando pasiones. En un contacto telefónico con Atlantic Devices, distribuidores en España de MetaCreations entre otros, nos confirmaron que todos los envíos procedentes de USA estaban vendidos y que la avalancha era tal que habían solicitado una ampliación urgente del stock. Toda esta Bryce-manía está justificada por una nueva versión que está llamada a ser la novedad más brillante del primer semestre del 99.



**Pocas novedades en la interfaz de usuario.**

## ¿Qué es Bryce?

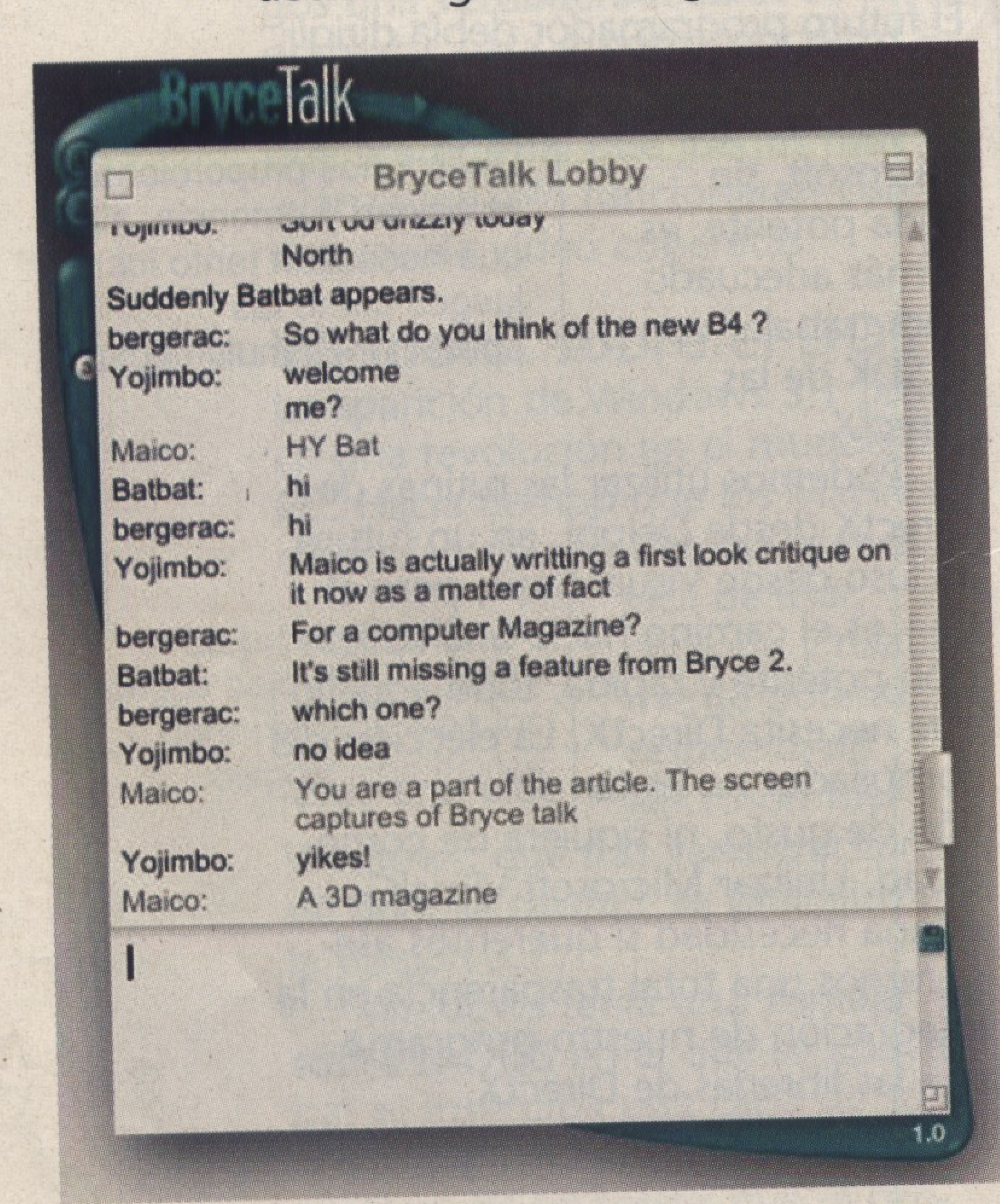
Tras su creación, Bryce se introdujo en el mundo del 3D por la puerta pequeña, la de la generación de escenarios. Cuando los modeladores e infografistas buscan programas de 3D se fijan más en el trabajo del modelado en sí que en lo que rodea al modelado. Bryce se desarrolló para generar este tipo de entornos en competencia (si es que la hubo) con aplicaciones como el VistaPro o World Builder. Pero Bryce no sólo es intratable en la generación de mundos tridimensionales sino que además se muestra muy capaz como herramienta de modelado 3D. De ahí que, en el mundo del arte digital, muchos de los autores más conocidos utilicen habitualmente esta herramienta.

## Al diseño por el diseño

En cuanto a la interfaz, Bryce no sólo destaca por sus capacidades, también ha introducido el concepto Kai de trabajo dentro del mundo 3D. Nada más arrancar cualquiera de las versiones de Bryce, cualquier referencia a nuestro sistema operativo desaparece y encontramos únicamente el interfaz de usuario del programa. Un elegante UI al más claro estilo de Kai: llamativos iconos 3D, ausencia de menús desplegables, curvas suavizadas, etc. Y es que la herencia del recién abducido Kai Krause está patente en cada menú. Kai se ocupó de formar un equipo de diseño que, aún hoy, crea escuela en cada aplicación o en cada Plug-in.

## Bryce at work

Sería muy sencillo además de obvio deciros que éste es el más potente de todos los Bryce, pero es que, comparativamente, es el que más novedades ha aportado y el que más potencia ha demostrado. Metacreations, consciente de que ésta es su más firme apuesta en 3D, ha chequeado el mercado en pos de los más avanzados progra-



**Saludos de los Bryce maniacs desde el Talk.**

mas 3D y ha realizado un resumen de lo mejor de cada uno de ellos, incluyendo todas esas características en esta nueva versión.

Por ejemplo, hay que señalar la increíble potencia de Four Seasons, un Plug-in para Photoshop y 3D Max capaz de generar todo tipo de cielos y definir gran multitud de variables como la fuga, las capas de nubes, etc. Pues bien, Bryce 4 incluye como novedad el Sky Lab o laboratorio de cielos que, básicamente, trabaja igual que el anterior pero con grandes diferencias, tanto en el cuidado diseño de su interfaz (Made in Metacreations) como por las posibilidades que nos brinda y de las que hablaremos en profundidad más adelante.

Además de estas prácticas novedades, hemos observado que las áreas más habituales del programa, y por tanto del trabajo 3D, se han potenciado e incluso adaptado a los estándares impuestos por los grandes dinosaurios del 3D. Así, se han incorporado algunas novedades que, como nos viene acostumbrando la firma californiana, sólo tienen ellos y muy pronto se verán copiadas. Pero esto lo veremos ampliado a continuación.

## Novedades

Más que entrar a fondo en el programa, con el que la mayoría ya estaréis de sobra familiarizados, vamos a hacer un repaso a fondo por las novedades que aporta esta nueva versión:

- *Sky Lab.* Se ha rehecho esta parte con el fin de obtener todo un laboratorio de cielos en el que podemos generar todo tipo de ambientes de una manera rápida e intuitiva. El control sobre los cielos y sobre las diferentes variables que nos brinda el programa es total. Además, gracias a la rápida previsualización, podemos ver casi en tiempo real los cambios realizados. Asimismo, para paliar la complejidad de los cambios, MetaCreations ha dotado a las diferentes variables




**Con la opción de exportar terrenos como mallas 3D podemos optimizar su número de polígonos.**

### Formatos compatibles

*- Importación de imágenes:*
*Bitmap ( .bmp)*
*Enhanced Metafile ( .emf)*
*Flashpix ( .fpx)*
*CompuServe GIF ( .gif)*
*JPEG ( .jpg)*
*Photoshop ( .psd)*
*Targa ( .tga)*
*TIFF ( .tif)*
*PICT ( Solo en Macintosh)*

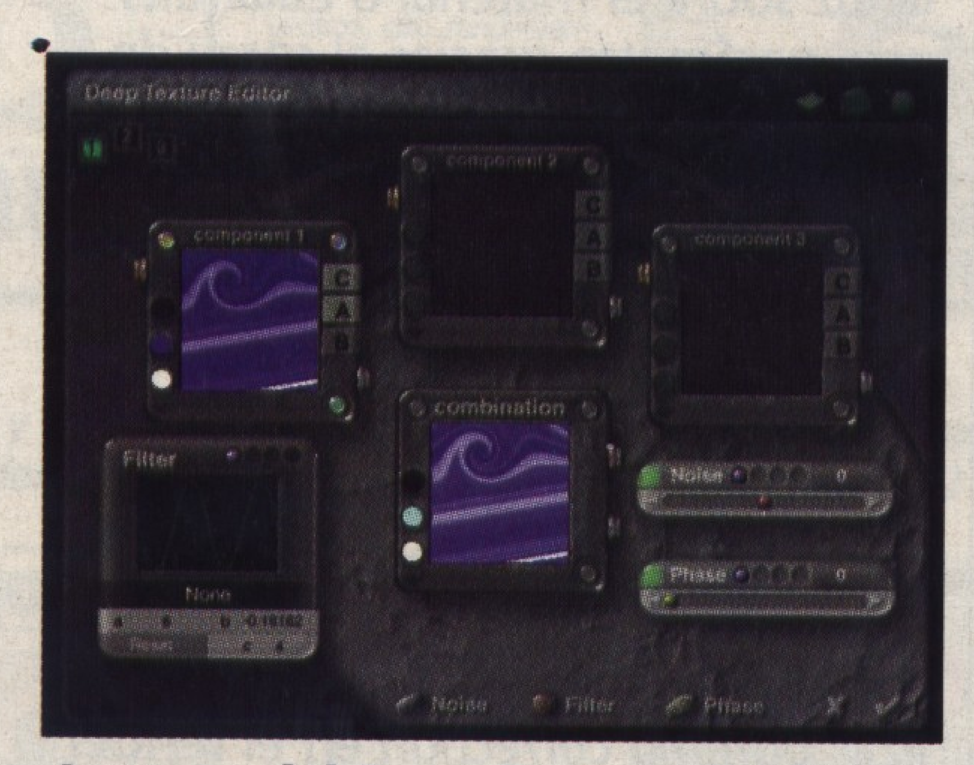
*- Exportación de imágenes. Bryce es capaz de exportar a todos los formatos de importación (excepto .emf) además de:*

*HTML ( .htm)*
*Quick Time VR Panorama ( .mov)*

*- Importación de Objetos:*
*TrueSpace ( .cob). Versiones superiores a la 4.0*
*VideoScape ( .vsa)*
*VRML1 ( .wrl)*
*LightWave ( .lwo y .lws)*
*Heightfield ( .hf)*
*USGS DEM ( .dem)*
*USGS SDTS ( .ddf)*
*Portable Greyscale Maps ( .pgm)*

*- Exportación de objetos. Bryce es capaz de exportar a todos los formatos con los que es compatible en importación además de:*
*RayDream Studio ( .rds)*
*AutoCad ( .dxf)*
*Infini-D 4.0 ( .id4)*
*Wavefront ( .obj)*

*Y seguro que a esta larga lista se le irán añadiendo nuevos formatos, entre ellos todos los que la casa de Carpintería nos ofrecerá en breves revisiones y apariciones.*




**Aspecto del Deep Texture Explorer.**

la posibilidad de ser animadas, gracias a lo cual será mucho más sencillo interactuar con los controles.

El *Sky Lab* está dividido en tres partes. La primera es *Sun & Moon,* donde determinamos si queremos que el cielo represente el día o la noche. Para ello podemos actuar sobre muchas variables. Intensidad y cantidad de estrellas (para definir si queremos una noche más o menos estrellada), intensidad y cantidad de cometas (gracias al cual podemos simular el paso de un cometa, una lluvia de estrellas o, y para esto es perfecto, simular paisajes extraterrestres). Podemos elegir entre ver o no la luna en el cielo así como su fase, brillo, etc. E incluso el efecto con el horizonte. También podemos controlar las estelas debidas a los brillos de sol o la luna, o anillos, su presencia e intensidad. En cuanto al sol, podemos definir también su presencia, color, posición e intensidad de su brillo.

- *Cloud Cover.* Como su nombre indica, en este apartado vamos a definir la presencia o no, así como el aspecto, de la cobertura de nubes en el cielo. Para ello Bryce 4 nos permite actuar con nubes de tipo estratos o cúmulos. Dentro de cada uno de estos tipos de nubes, podemos definir los colores de éstas, su altura, tamaño, forma, dirección, su frecuencia, amplitud, etc. Y , como no, podemos fijar o no las nubes a la cámara, fijarlo a la imagen, generar mapeados esféricos, etc.

- *Atmosphere.* Es, quizás, el mando central del cielo, dado que en este apartado podemos controlar las variables *Haze* y *Fog* y el resto de controles principales de los cielos. Podemos controlar el efecto de perspectiva y definir el color del cielo. Además podemos incorporar efectos atmosféricos como, por ejemplo, un arco iris, y definir su radio y opacidad. Asimismo podemos definir el efecto de mezcla de la atmósfera con el sol a partir del color y luminancia del efecto.

## Compatibilidad

No olvidemos que Metacreations es una firma en constante proceso de creación. Mes a mes nos sor-




**Control total sobre nubes en el Sky Lab.**


Ventana de importación de imágenes.

prende con nuevos programas o nuevas versiones de programas ya existentes (como muestra os presentamos dos nuevas versiones de programas tan importantes como Ray Dream Studio y Poser) lo que desemboca en una gran cantidad de nuevas extensiones y formatos a reconocer. Tanto es así que hemos preferido que evaluéis por vosotros mismos lo que decimos y, para ello, os adjuntamos un cuadro en el que podréis ver una breve lista de formatos compatibles.

## Bryce y la web

Como ya viene siendo habitual en algunos programas de la casa, Bryce incluye la posibilidad de chatear con otros usuarios registrados a través de *Bryce Talk*. Los primeros pasos de esta herramienta los vimos en Painter, la herramienta de arte digital de la casa californiana Fractal Design (una de las fundadoras de MetaCreations). Ahora se ha reformado adaptando su diseño a las tendencias estéticas impuestas por Kai, dando como primer fruto el *Sopa Talk,* incluido en la versión 2 del popular programa de retoque y gestión fotográfica. Por ahora muy poca gente está conectada, dada la novedad del programa, pero MetaCreations ya ha comenzado a realizar sesiones en los diferentes continentes. En algunas imágenes adjuntas podéis ver la conversación que mantuvimos con usuarios de Silicon Valley, Dublín y Nueva York.

Por otro lado, Bryce ha incorporado la posibilidad de incluir hipervínculos dentro de sus paisajes gracias a lo cual, usando los módulos de exportación, podemos generar *Web Maps* en HTML así como escenarios virtuales panorámicos en formato QuickTime VR.

Ni que decir tiene que Bryce nos brinda la posibilidad de exportar en formato Metastream nuestros trabajos. Para aquellos que no lo conozcáis, MetaStream es una potente herramienta de VR que nos permite interactuar con los objetos de una manera hasta ahora desconocida y desde archivos fácilmente transferibles a través de la red.

## Mejoras en el módulo generador de terrenos

Metacreations no podía dejar de potenciar uno de los pilares básicos del programa. Para ello, ha incorporado más de 20 nuevos modelos fractales para la generación de terrenos, así como ha aumentado su capacidad con un nuevo módulo de exportación que nos permite definir el tipo de malla sobre la que queremos exportar e incluso optimizar este proceso para reducir el número de polígonos.

## Mejoras en el módulo de animación

Otra de las novedades de Bryce es la mejora en el trabajo de animación. Ahora podemos visualizar inmediatamente nuestra animación en la ventana de preview sin tener que rendear la escena. Además podemos rendear sólo algunas partes de la escena para evaluar la incidencia de la luz sobre el material o cualquier otro parámetro gracias al spray de render. También podemos generar previos mediante *Thumbnails,* o pequeñas imágenes entre las que nos podemos mover y trabajar gracias a su interfaz tipo Story Board.

## Mejoras en el Render

Metacreations ha ampliado las posibilidades de render del programa añadiendo la posibilidad de utilizar *gamma correction* en el render así como *dithering* de hasta 48 bits. Asímismo el aspecto de la iluminación, que tanta importancia cobra en el render, nos abre un nuevo abanico de posibilidades al aumentar considerablemente las opciones de control sobre los parámetros de iluminación.

## Y mucho, mucho más...

Sin duda, son innumerables las novedades de esta nueva versión, y ésa es precisamente una de las notas negativas. El lanzamiento oficial en España se produjo hace apenas unos días y no hemos tenido tiempo de trabajar con él lo suficiente como para poder ofreceros una visión más global de las nuevas virtudes del programa. Nuestra idea es que seáis los primeros en conocerlo, por ello queríamos que en este número tuvieseis el primer avance de un programa que va a darnos de que hablar a todos y del que seguramente tendréis cumplida información en próximos números.

## La nota negativa

Sólo hemos observado la misma carencia que en las versiones anteriores: la falta de una herramienta texto. Y es que una gran parte del trabajo de animadores e infografistas se basa en la animación de logotipos, así como en la incorporación de efectos impactantes a videos y animaciones (muchas de ellas basadas en aparición de textos o datos numéricos). Por ello, creemos que es una lástima que haya que importar el texto desde otra aplicación como malla u objeto.

Michel Chelton

## Ficha técnica Bryce 4

**Fabricante:** MetaCreations
**Distribuidor:**
Atlantic Devices
Caputxins, 58
08700 Igualada (BCN)
Tlf: 93.804.07.02
93.804.01.60
e-mail: atlantic@lander.es
**Más información:**
www.metacreations.com
www.atlanticdevices.com
**Precio:** 49.500 pts. + I.V.A.

## Requisitos mínimos de Bryce 4

**Windows:**
Procesador Pentium
Windows 95/98 o NT 4
32 Mb de RAM bajo Windows 95/98
64 Mb de RAM bajo NT 4 (recomendados)
75 Mb de espacio libre en disco para la instalación mínima
Tarjeta de color de 24 bits
Lector Cd-Rom.
Módem con conexión a Internet (para Bryce Talk).

**Macintosh:**
Procesador Power Macintosh
S.O. 7.5.5 o superior
32 Mb de RAM
75 Mb de espacio libre en disco para la instalación mínima
Tarjeta de color de 24 bits
Lector Cd-Rom.
Módem con conexión a Internet (para Bryce Talk).



Créditos de Bryce 4, año I después de Kai.

# CATÁLOGO DE LIBROS DE PRENSA TÉCNICA

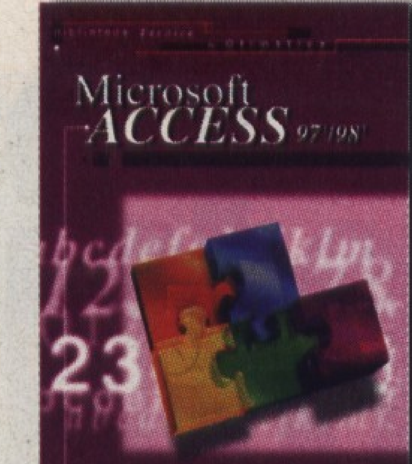
### Cómo pasar 10 minutos divertidos en el Chat




La comunicación en tiempo real hablada y escrita, la videoconferencia y el IRC dejarán de tener secretos para ti con esta obra dedicada a hacer más divertido Internet cuando chateamos.

**2.995 ptas. Incluye CD-ROM.**

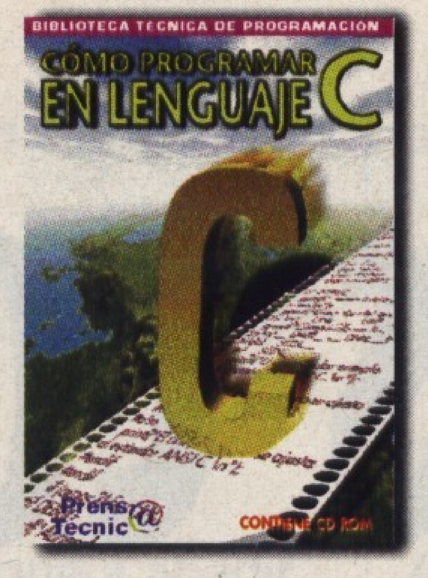
### Microsoft Access 97-98




Microsoft Access, una de las aplicaciones más populares hoy en día, que ha logrado consolidarse como un auténtico estándar entre los gestores de bases de datos para entornos gráficos como es el caso de Windows.

**2.995 ptas. Incluye CD-ROM.**

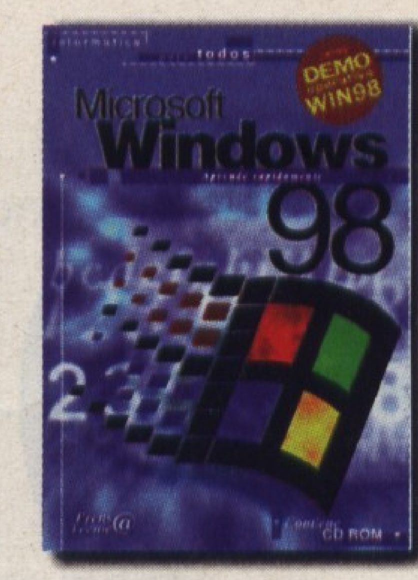
### Cómo Programar en Lenguaje C




El lenguaje C es el modelo de programación por excelencia, el más utilizado para todo tipo de aplicaciones. Este libro le introducirá en la programación C de una manera clara y sencilla.

**2.995 ptas. Incluye CD-ROM.**

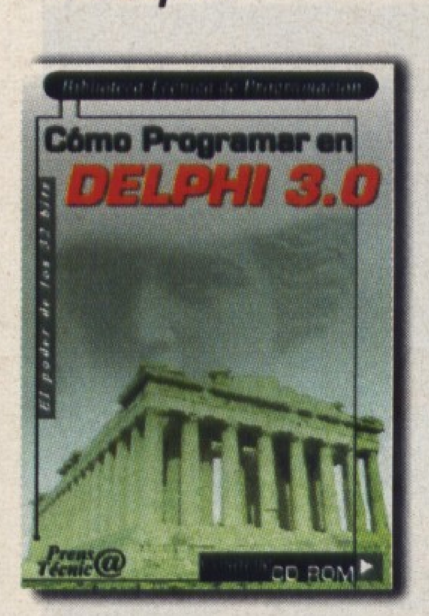
### Cómo trabajar con Windows 98




Descubra todas los avances del nuevo sistema operativo de Microsoft y domine en poco tiempo todas las formas de trabajar con él. Incluye un CD-Rom con una demo completamente operativa del programa.

**2.995 ptas. Incluye CD-ROM.**

### Cómo Programar en Delphi.




Delphi se puede considerar como uno de los entornos de programación visual más poderosos y a la vez más fáciles de utilizar y aprender. Esta obra está orientada a nivel principiante e intermedio.

**2.995 ptas. Incluye CD-ROM.**

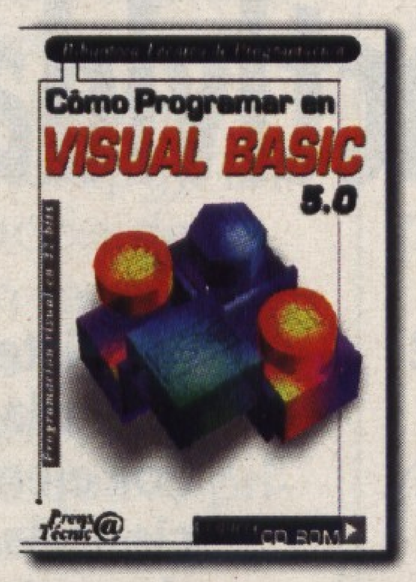
### Programación Gráfica para PC




Es una obra dedicada a todos aquellos que quieran adentrarse en el apasionante mundo de la programación visual. Se puede aprender, de una forma rápida y sencilla, a explotar todas las capacidades gráficas de su ordenador.

**2.995 ptas. Incluye CD-ROM.**

### Cómo Programar en Visual Basic




Con Visual Basic 5.0 creará aplicaciones Windows rápidamente, trabajando con una de las herramientas de desarrollo más potentes del mercado. Con este libro conocerá sus propiedades y conceptos básicos de programación.

**2.995 ptas. Incluye CD-ROM.**

### Cómo Programar tus Propios Juegos




Introduce al lector de forma sencilla en el mundo de la programación de videojuegos, utilizando las técnicas más avanzadas en este campo recorrido por la historia de los videojuegos.

**2.995 ptas. Incluye CD-ROM.**

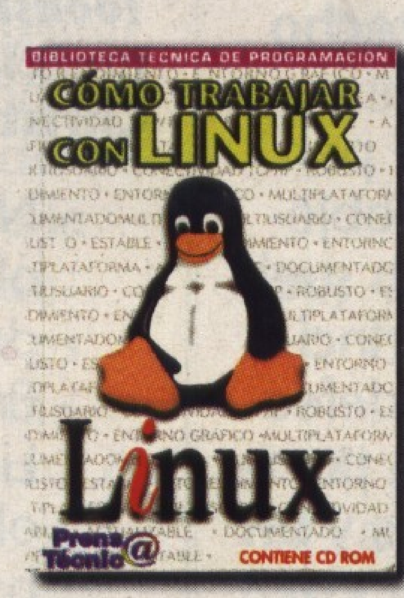
### Cómo trabajar con Corel Draw 8




Herramienta de trabajo que puede ser utilizada tanto a nivel profesional como particular. Manual práctico con el que acceder a las múltiples posibilidades de Corel-Draw.

**2.995 ptas. Incluye CD-ROM.**

### Cómo trabajar con Linux




Explica cómo comenzar a usar Linux y será de utilidad para sacar más partido a su instalación. El sistema ofrece multitarea, potente entorno gráfico, alto rendimiento y conectividad.

**2.995 ptas. Incluye CD-ROM.**

### Directorio Fundamental de Internet




Directorio Fundamental de Internet es una completa guía, en la que podrá encontrar las direcciones más útiles al llevar a cabo la compleja tarea de navegar buscando un tema concreto.

**2.995 ptas. Incluye CD-ROM.**

### Manual Técnico AutoCAD 14




Descubre la última versión de AutoCAD 14, un programa de dibujo de propósito general, mayoritamente difundido en el ámbito del diseño asistido por ordenador.

**2.995 ptas. Incluye CD-ROM.**

### Manual Técnico Lightwave




Introduce al lector en el mundo del modelado de Lightwave de una forma sencilla y práctica: creación de efectos atmosféricos, animaciones, etc. Incluye ejemplos prácticos.

**2.995 ptas. Incluye CD-ROM.**

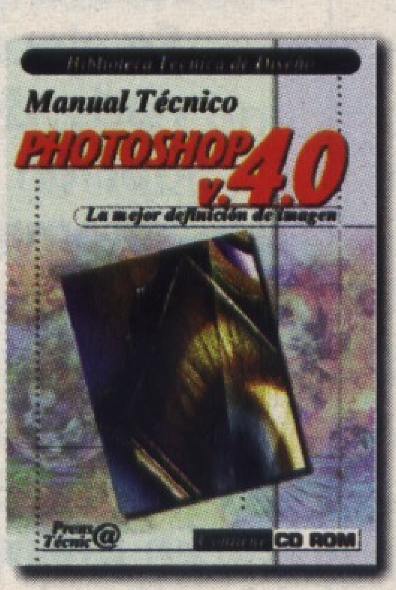
### Música y FX para videojuegos




Medios Técnicos necesarios para la elaboración de la banda sonora de un videojuego, así como los efectos de sonido. Todo ello acompañado de una serie de ejercicios prácticos que el lector podrá ir realizando a medida que avanza el libro.

**2.995 ptas. Incluye CD-ROM.**

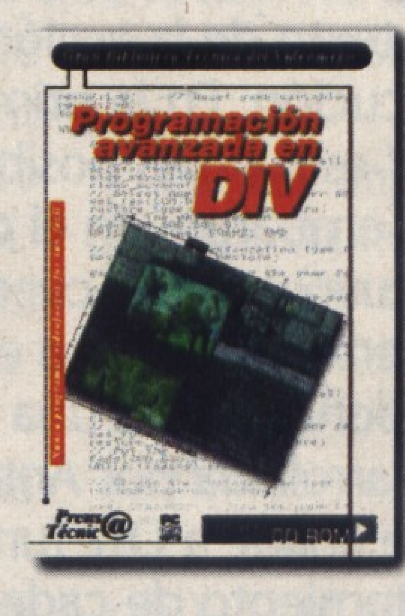
### Manual Técnico PhotoShop 4.0




Photoshop es el software de retoque fotográfico por excelencia y el programa más utilizado por los profesionales del diseño. Esta obra permitirá al lector adentrarse en el mundo de la imagen digitalizada, su tratamiento, etc.

**2.995 ptas. Incluye CD-ROM.**

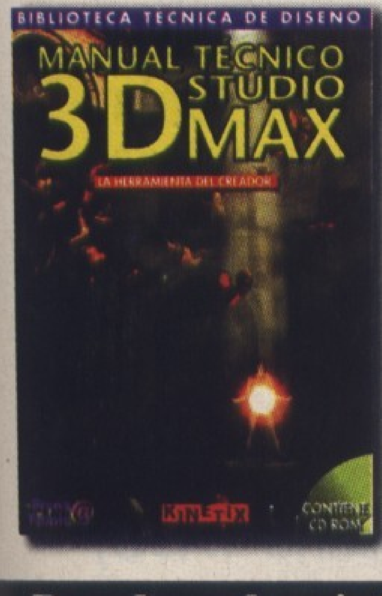
### Programación avanzada en DIV




Conozca los últimos avances en programación bajo DIV Games Studio, con ejemplos, trucos, modos de trabajo y una galería de direcciones web y canales de IRC sobre el mundo DIV.

**2.995 ptas. Incluye CD-ROM.**

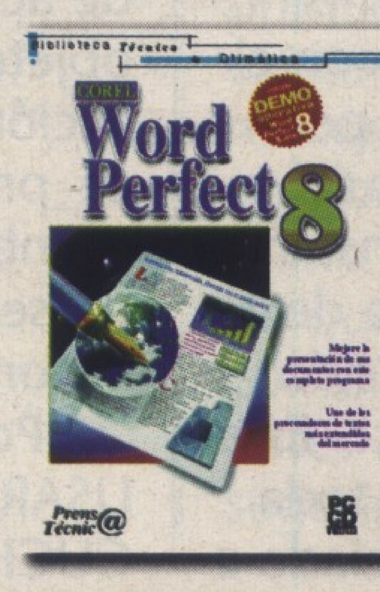
### Manual Técnico 3D Studio Max




La obra definitiva tanto para el usuario novel como para los ya iniciados, que lograrán dominar esta herramienta en poco tiempo. Incluye CD-Rom repleto de utilidades, modelos y texturas para trabajar con 3D Max.

**2.995 ptas. Incluye CD-ROM.**

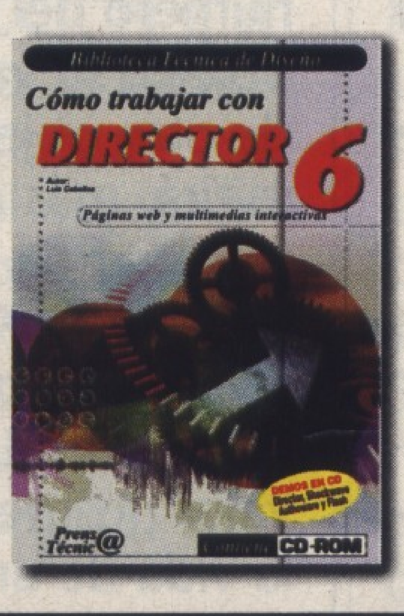
### Manual Práctico de Corel WordPerfect 8




El dominio de uno de los programas de tratamiento de texto está a tu alcance con este libro que repasa todas las funciones. Nunca resultó tan sencillo sacarle el máximo partido a WordPerfect.

**2.995 ptas. Incluye CD-ROM.**

### Cómo trabajar con Director 6




Páginas web y creación multimedia con este libro, que recorre uno a uno los aspectos fundamentales de Director 6, un programa que permite trabajar con sonidos, animaciones, texto, vídeo digital, etc.

**2.995 ptas. Incluye CD-ROM.**



**Puede adquirir estos productos llamando al nº de teléfono 91 304 06 22 o en los distribuidores oficiales indicados abajo.**

## TRIBUIDORES

**LLA Y LEÓN:**
ES GARCÍA LIBROS, S.L. C/ Pintores, 5. s de la Reina, 37184. Salamanca. 23 23 02 06, fax: 923 25 31 17.

**NCIA-CASTELLÓN:**
O GENERAL DE LLIBRERIA, S.L. Avd. Ausias 34 Bajo. Tavernes Blanques. 46016 Valencia. 61 85 18 28, fax: 961 85 88 91.

**UNYA Y BALEARES:**
BUCIONES PRÓLOGO, S.A. C/ Mascaro, 032 Barcelona. 34 56 20 00, fax: 934 56 95 06.

**D Y CASTILLA LA MANCHA:**
FORMA, S.A. C/ Abtao, 25 Nave B. Madrid. 15 01 47 49, fax: 915 01 48 99.

**ALICANTE-MURCIA-ALBACETE**
GAIA LIBROS, S.L. C/ Sagitario, 7-9. 03006, Alicante
Telf: 965 11 05 16, fax: 965 11 41 26.

**GALICIA**
GRIAL LIBROS, S.L. Pol. Ind. Tambre, Via Diesel, 4. 15890 Santiago de Compostela.
Telf: 981 58 97 54, fax: 981 57 59 04.

**ARAGÓN-LA RIOJA-SORIA**
ÍCARO DISTRIBUIDORA, S.L. Pol. El Plano, C/ A,39, Mª de Huerva. 50430 Zaragoza.
Telf: 976 12 63 33, fax: 976 12 64 95.

**ANDALUCÍA ORIENTAL**
NADALES LIBROS, S.A. Camino Bajo S/N, Armilla. 18100 Granada.
Telf: 958 55 08 03, fax: 958 57 15 56.

**PAIS VASCO-NAVARRA**
TOMÁS ELLACURIA, S.A. Pol. Legizamon, C/ Nervion, 1 bajo. 48450 Etxebarri, Vizcaya.
Telf: 944 26 12 50, fax: 944 26 12 88.

**ASTURIAS-SANTANDER**
NORTE PROMOCIONES, S.A. C/ Pérez Ayala, 10. 32208 Gijón.
Telf: 985 14 70 30, fax: 985 38 88 14.

**EXTREMADURA**
UNIDISA, S.A. Pol. Ind. El Nevero, Complejo La Mar, Nave 10. 06006 Badajoz.
Telf: 924 27 51 92, fax: 924 27 56 01.

**ISLAS CANARIAS**
TROQUEL DISTRIBUCIONES. Urb. Montaña Blanca, Parcela 31A, Arucas. 35415 Gran Canaria.
Telf: 928 62 17 80, fax: 928 62 17 81.

**ANDALUCÍA OCCIDENTAL**
DYD/UNIDISA, S.A. Pol. El Pino, Parcela 8E Nave 21-22. 41016 Sevilla.
Telf: 954 51 63 33, fax: 954 25 50 98.

**DE VENTA EN LIBRERÍAS**

# Acciones con objetos

## Creando la interactividad

**Con la interfaz gráfica que definimos en el artículo anterior y con gran parte del corazón de nuestro juego hecho, vamos a definir los objetos del escenario y cómo aplicar las acciones del menú a cada uno de ellos.**

Como ya vimos en el anterior Divmanía, el constructor gramatical será el encargado de generar la línea de acciones de acorde con las opciones del menú y los objetos seleccionados. Para conseguir esto, se debe crear la cadena de texto de la acción en tiempo de ejecución. En la primera versión de DIV esto no es posible directamente. Sin embargo, en el desarrollo de DIV se tuvo muy en cuenta que debía ser abierto y, gracias a esta característica, disponemos de una librería ascii.dll desarrollada por Antonio Marchal que añade a DIV funciones para el tratamiento de cadenas.

### Definiendo

DIV nos da la posibilidad de detectar colisiones entre procesos; esto nos da una gran ventaja a la hora de preparar la forma en que vamos a interactuar con los objetos. Puesto que el puntero del cursor es un proceso, nos basta con hacer que cada objeto lo sea también. Cada objeto será un proceso con el que colisionará el cursor. El proceso debe detectar esta colisión y actuar en consecuencia. Cuando se produzca una colisión, se debe tener en cuenta que:

DIV nos permite detectar colisiones entre procesos, lo cual es una gran ventaja a la hora de preparar la forma en que vamos a interactuar con los objetos

**1. No se pulsa ningún botón del ratón:**
- Si no existe ninguna acción seleccionada, mostramos descripción del objeto en la línea de acciones.
- Si existe alguna acción seleccionada, la complementamos. Por ejemplo, si la acción seleccionada es coger, en la línea de acciones aparecerá coger objeto.

**2. Se pulsa el botón derecho del ratón:**
- Se seleccionará la acción por defecto a realizar con el objeto. La opción por defecto suele ser mirar, al menos en principio. Otras opciones por defecto pueden ser abrir, para puertas, hablar, para personas y un largo etcétera.

**3. Se pulsa el botón izquierdo del ratón:**
- Se ejecutará la acción seleccionada. Si queremos que la opción por defecto se ejecute automáticamente, cuando se detecte que se pulsó el botón derecho, se forzará a *TRUE* el botón izquierdo.

En primer lugar vamos a crear identificadores de los procesos que nos permitan distinguirlos. Las constantes id_XXXX (donde las X son el número del objeto) identifican el objeto; su valor es irrepetible con el de otro objeto. Este valor nos permitirá, más adelante, reconstruir una partida guardada. Las constantes to_XXXX definen el estado de un objeto, por ejemplo, si ha sido usado, si está lleno o vacío, etc. Este valor nos permitirá hacer que toda la vida de un objeto vaya asociado a un único proceso. De momento estas constantes no tienen mucho sentido y no serán realmente útiles hasta futuras lecciones.

Sí que es muy importante que cada objeto tenga a su disposición una serie de variables. Esto lo conseguimos a través de la cláusula LOCAL. Las variables a las que debemos tener acceso son:
- NumID : entero. Número identificador del objeto (id_XXXX).
- Estado : entero. Estado actual del objeto (to_XXXX).




***Toonstruck* es una aventura que mezcla actores de la talla de Christopher Lloyd.**

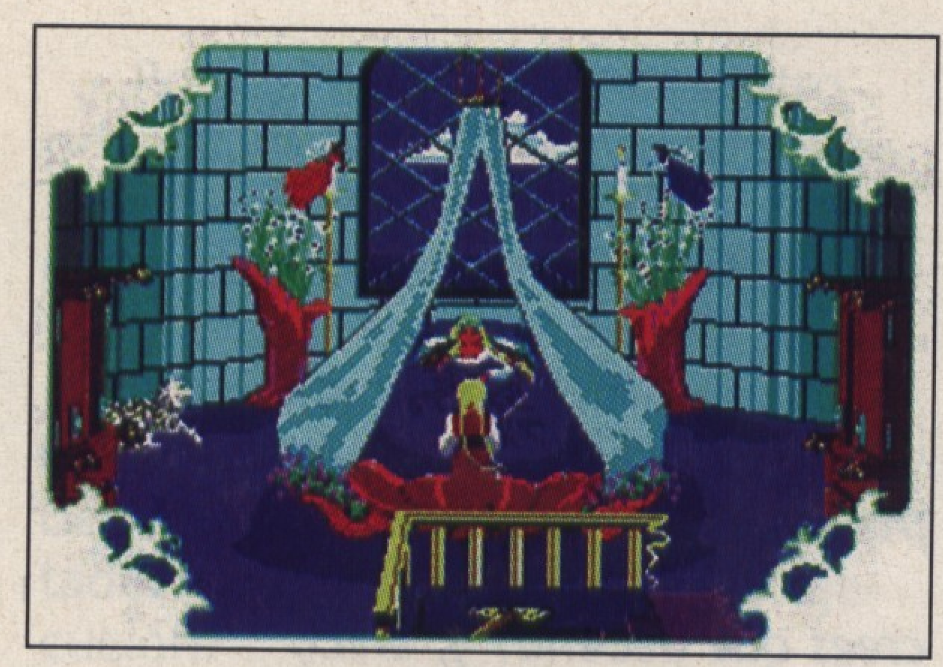
- Nombre : cadena. Nombre del objeto, visualizable en la línea de acciones.
- EnlacesPrimarios[10] : cadena. Lista de enlaces cuando es el objeto primario. Ejemplo Ir a b; a es el enlace y b el objeto primario.
- EnlacesSecundarios[10] :cadena. Lista de enlaces cuando es el objeto secundario. Ejemplo: Usar a con b; con es el enlace, a el objeto primario y b el secundario.
- Ejecuta : lógico. Indica al objeto si debe ejecutar la acción o no.

Nuestra meta es mejorar el proceso pon_accion, encargado de escribir la línea de acción. Para construir esta línea tendremos los siguientes elementos básicos: texto de acción, id objeto primario, id objeto secundario. Así, la línea de acción será: texto_acción + enlace_primario[objeto_primario] + nombre_objeto_primario + enlace_secundario[objeto_secundario] + nombre_objeto_secundario

Por ejemplo, para una acción USAR, donde el objeto primario es CUCHARA y el secundario VASO,



***Ring* es una de las más avanzadas aventuras, lejos de los antiguos títulos conversacionales.**



***Kings Quest*, una de las sagas de aventuras que ha logrado inmortalizar a Sierra.**

tendríamos: USAR + (nada) + CUCHARA + CON + VASO. También crearemos un procedimiento que determine quién debe ejecutarse.

## Implementando

Después de haber definido las constantes para los objetos, creamos un bloque LOCAL para definir las variables comunes que tendrán todos nuestros objetos, seguido de la llamada a la importación de la librería ascii.dll. Lo primero que debe hacer nuestro programa es comprobar que la versión de la librería es la adecuada (versión 2 ó superior).

```
o_primario   = 11;
o_secundario = 13;

// ID objetos
id_diana          = 105;

// Estado objetos
to_diana          = id_diana;

LOCAL
    NumID               = o_none;
// ID objeto
    Estado              = o_none;
// Estado objeto
    Nombre              = "";
    EnlacesPrimarios[10] = "", "",
"", "", "", "", "", "", "", "";
    EnlacesSecundarios[10] = "", "",
"", "", "", "", "", "", "", "";
    Ejecuta             = FALSE;

    Import "ascii.dll";

BEGIN
```



**El aspecto gráfico debe ser portentoso para que pueda impactar en el público.**

```
    // Se comprueba la correcta
versión de la DLL
    IF (ascii_version() < 2 )
        exit("Necesario fichero ascii.dll
versión 2", 1);
    END;
```

Parte del nuevo bloque a añadir antes y después del BEGIN principal.

Cargaremos un nuevo fichero de fuentes (verde.fnt) para el proceso habla y el fichero gráfico donde tenemos el dibujo de una diana para este ejemplo (objetos.fpg). En el bloque de la inicialización de la pantalla escribiremos un texto en blanco para id_t. Esto es imprescindible para el funcionamiento del procedimiento habla, ya que lo primero que intenta es borrar un texto; por tanto debe existir un texto que pueda borrar.

```
PROCESS habla(texto);
BEGIN
    delete_text(id_t);
    id_t = write (f_verde, 320,
100, 4, texto);
END;
```

El proceso habla será mejorado más adelante, por el momento nos basta este sencillo código.

Seguidamente llamaremos al procedimiento objeto. Del control de menú de opciones quitaremos el comentario de sel_objeto(o_none, FALSE). Ya contamos con esta función, al igual que haz_accion, teniendo que quitar el comentario que le pusimos en el proceso sel_accion.

## Los nuevos procesos

El proceso habla no requiere especial atención por el momento. Nuestro proceso pon_accion() ha cambiado considerablemente. El primer lugar coge la ACCION y, si existe el objeto primario, la combina con el enlace primario y el nombre del objeto. Si el objeto primario está seleccionado, en el caso de que exista objeto secundario, combina la línea de acción de antes con el enlace secundario del segundo objeto y su nombre.

```
PROCESS pon_accion();
PRIVATE
    t;
BEGIN
    LOOP
        t = Accion.t_accion;
        IF (Accion.id_obj_primario)
            t = ascii_add(t,
(Accion.id_obj_primario).EnlacesPrim
arios[Accion.id_accion]);
            t = ascii_add(t,
(Accion.id_obj_primario).Nombre);
        END;
        // )Seleccionado el objeto
primario?
            IF (Accion.nivel_obj ==
o_none)             // No -> bórralo
de acción
                Accion.id_obj_primario =
o_none;
            ELSE
// Si -> busca secundario
                IF (Accion.id_obj_secunda-
rio)
                    t = ascii_add(t,
(Accion.id_obj_secundario).EnlacesSe
cundarios[Accion.id_accion]);
                    t = ascii_add(t,
(Accion.id_obj_secundario).Nombre);
                END;
            END;
            Accion.id_t_accion =  write
(f_sistema, 320, 370, 4, t);
            FRAME;

delete_text(Accion.id_t_accion);
    END;
END;
```

Proceso pon_accion después de los nuevos cambios.

Como procedimiento nuevo tenemos sel_objeto y des_objeto. El primero nos servirá para seleccionar objetos y el segundo para anular las selecciones. Como parámetros de sel_objeto se le pasa el id del objeto y TRUE o FALSE para afirmar si el objeto se selecciona o no. En primer lugar comprueba que idobj no sea nulo (o_none); de lo contrario, se asegura de que no haya seleccionado ningún objeto. Si idobj es un identificador de objeto válido (no es nulo), comprueba el nivel de la acción (Accion.nivel_obj). Si el nivel de la acción es objeto primario, trata al nuevo objeto como el primero; si es secundario, lo trata como al segundo. Si como parámetro la pasamos *TRUE*, selecciona al objeto, cambiando el nivel (si era primario pasa a secundario, si ya era secundario queda igual). Seguidamente ejecuta la acción llamando a haz_accion.

**Es muy importante que cada objeto disponga de una serie de variables. Esto lo conseguiremos a través de la cláusula LOCAL**

El procedimiento des_objeto deselecciona todos los objetos seleccionados.

```
    PROCESS sel_objeto(idobj,
select);
    BEGIN
```

### Notas sobre los ficheros fuentes

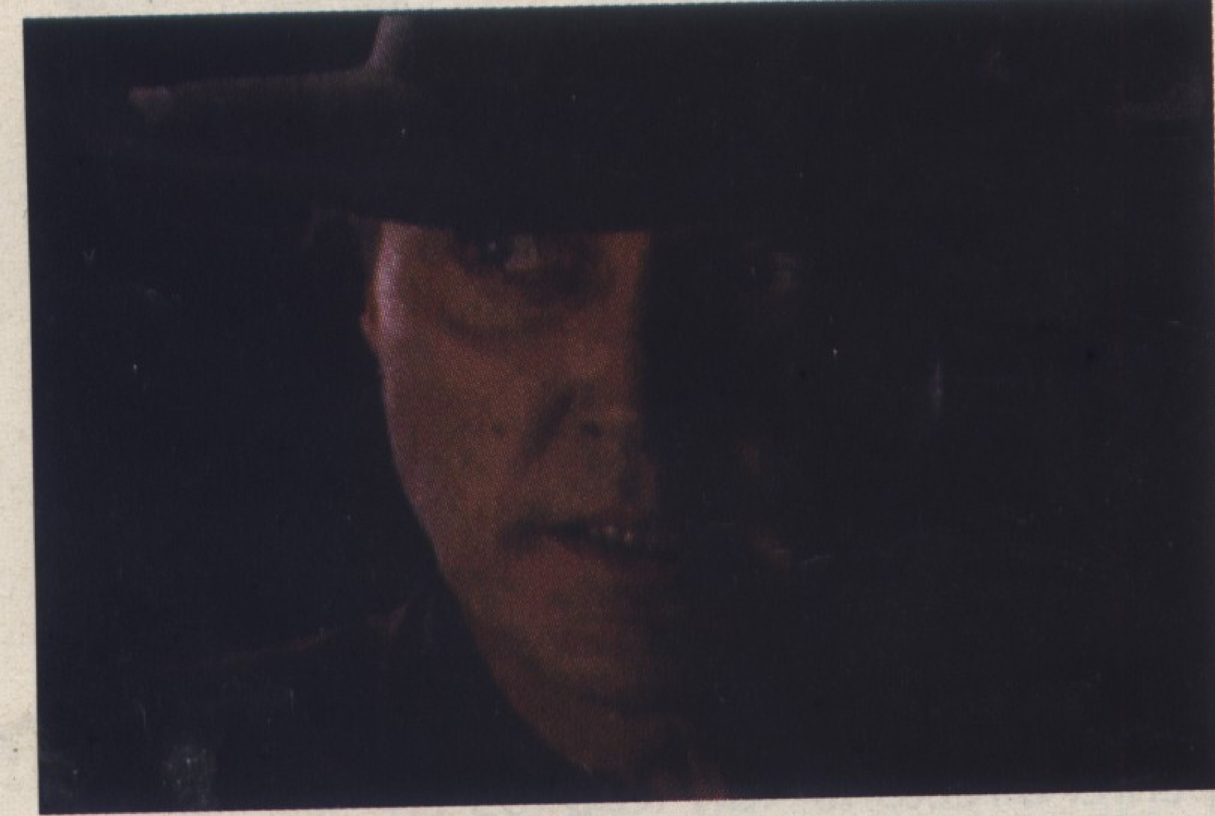
Los ficheros fuente del CD están preparados para ser instalados en el directorio \div\cag\03 en la unidad donde tengas instalado DIV Games Studio.

```
IF (idobj == o_none)
// ninguno -> borra accion
    Accion.Seleccionada =
FALSE;
    Accion.id_accion =
o_none;
    Accion.id_obj_primario =
o_none;
    Accion.id_obj_secundario =
o_none;
    Accion.nivel_obj =
o_none;
  ELSE
    IF (Accion.nivel_obj ==
o_none)          // 1' nivel ->
obj. Primario
      Accion.id_obj_primario =
idobj;
    ELSE
// Es objeto secundario
    IF ((Accion.id_obj_primario <>
idobj) &&
      (Accion.id_accion <> o_none))
// Hay opción activa -> vale obj. Sec.
      Accion.id_obj_secundario = idobj;
      END;
    END;
    IF (select)
      IF (Accion.nivel_obj ==
o_none)   Accion.nivel_obj = o_pri-
mario;
      ELSE
        Accion.nivel_obj =
o_secundario;
      END;
      haz_accion();
    END;
  END;
END;

PROCESS des_objeto();
BEGIN
  sel_objeto(o_none, FALSE);
END;
```

Código de los nuevos procesos sel_objeto y des_objeto

El proceso haz_accion es simple en realidad. Se asegura de que exista una acción y un objeto primario seleccionado. En este caso, fuerza a ejecutarse al objeto secundario si existe, si no, al objeto primario.



**Cada vez es más habitual que aparezcan grandes actores como Christopher Walken.**



**La abundancia de vídeo es una de las características de las aventuras modernas.**

```
PROCESS haz_accion();
BEGIN
  // Si opción seleccionada y
objeto seleccionado, ejecuta
  IF ((Accion.Seleccionada) &&
(Accion.id_obj_primario))
    // Si hay secundario, ejecú-
talo
    IF (Accion.id_obj_secundario)
      (Accion.id_obj_secunda-
rio).Ejecuta = TRUE;
    ELSE
      // No hay secundario, si
hay primario ejecútalo
      IF (Accion.id_obj_primario)

(Accion.id_obj_primario).Ejecuta =
TRUE;
      END;
    END;
  END;
END;
```

El proceso haz_accion es el encargado de obligar a ejecutarse al objeto adecuado.

Sería una locura intentar recoger todas las combinaciones posibles por cada objeto. Por eso sólo recogeremos en cada objeto las combinaciones de acciones que realmente tienen importancia con el objeto, dejando el resto a un proceso comodín. Este proceso comodín es acciones(accion_id), donde pasamos como parámetro el id de la acción. Si como id_accion pasamos o_none, se usará la accion actualmente seleccionada. Normalmente pasaremos como parámetro o_none, pero tenemos la posibilidad de tratar una acción como si de otra se tratase.

```
PROCESS acciones(accion_id);
PRIVATE
  n;
  ac;
BEGIN
  // Si se especificó una acción,
se hace según ella, si no, se toma
  // la acción seleccionada
  IF (accion_id == o_none)
    ac = Accion.id_accion;
  ELSE
    ac = accion_id;
  END;

  // Generamos un n1 aleatorio
que debe ser redondeado
  n = rand(0, 30);
  SWITCH (ac)
    CASE o_none:  END;
    CASE o_Ir_a:  END;
    CASE o_Hablar:
      SWITCH (n mod 2)
        CASE 0: habla("Sería
perder el tiempo"); END;
        DEFAULT: habla(")Tú
crees que debería?"); END;
      END;
    END;
    CASE o_Mover:
      SWITCH (n mod 3)
        CASE 0: habla("No
alcanzo"); END;
        CASE 1: habla("Parece
muy pesado"); END;
        DEFAULT: habla("No
creo que deba hacer eso"); END;
      END;
    END;
    CASE o_Mirar:
      habla("No veo nada en
especial");
    END;
    CASE o_Coger:
      SWITCH (n mod 4)
        CASE 0: habla("No lo
necesito"); END;
        CASE 1: habla("Podrían
echarlo en falta"); END;
        CASE 2: habla("Es muy
pesado"); END;
        CASE 3: habla("Mejor
dejarlo ahí"); END;
        CASE 4: habla("No lo
necesito"); END;
        DEFAULT: habla("No
puedo llevarlo"); END;
      END;
    END;
    CASE o_Dar:
      habla("Mejor no");
    END;
    CASE o_Abrir:
      habla("Parece no abrirse");
    END;
    CASE o_Cerrar:
      habla("Parece no cerrarse");
    END;
    CASE o_Usar:
      SWITCH (n mod 4)
        CASE 0: habla("No se
que hacer con eso"); END;
        CASE 1: habla("Parece
no funcionar"); END;
        CASE 2: habla("Hacer
eso no tiene sentido"); END;
        DEFAULT: habla("Mejor
no"); END;
      END;
    END;
    DEFAULT:
      habla("No entiendo lo
que quieres hacer");
    END;
  END;
  des_objeto();
END;
```

Proceso comodín para las acciones.

Por último, tenemos nuestro proceso objeto. Como parámetros se le pasan las coordenadas x e y junto con el fichero que contiene los gráficos. En primer lugar se da valor a su id, estado, nombre y enlaces. Dentro del bucle principal del proceso distinguimos tres apartados: movimiento, colisión y ejecución.

Para el ejemplo hemos creado un movimiento para que el objeto pueda desplazarse con las teclas q, a, o y p.

El apartado colisión es genérico para todos los objetos. En cuanto se detecta una colisión con el ratón, la variable Hubo_colision pasa a verdadero. Si se pulsa el botón derecho del ratón se selecciona la acción por defecto (o_mirar) y se fuerza al botón izquierdo del ratón a ser verdadero (para que se ejecute la acción por medio de haz_accion) inmediatamente. Si se pulsa el botón izquierdo del ratón y no hubo selección, se ejecuta a sí mismo (poniendo Ejecuta a verdadero) y marca que ya hubo selección. Si se pulsó el botón izquierdo pero ya hubo selección, se selecciona el objeto.

Si no hay colisión, pero la hubo antes, se desecha el objeto secundario. Esto es así porque podremos seleccionar un objeto primario y después buscar un secundario, pero en cuanto seleccionamos un secundario, la acción se ejecuta.

En el apartado ejecuta, comprobamos la acción seleccionada. En el caso de que sea o_none, se selecciona el objeto (esperando a seleccionar más tarde la acción). En el caso de que sea otra acción, actuamos en consecuencia. En nuestro ejemplo sólo tratamos la acción o_Mirar, llamando a la función comodín en el resto. No hay que olvidar dejar de seleccionar el objeto después de ejecutarlo y poner Ejecuta a falso para que no se active sucesivamente.

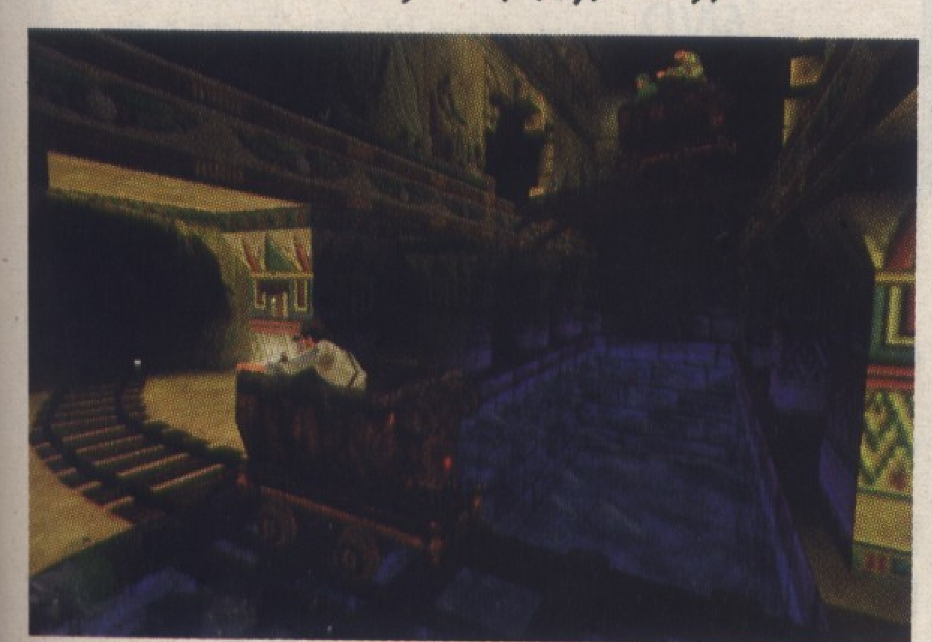
En nuestro ejemplo, partimos de una cosa que conforme la vamos mirando averiguamos lo que es. Presta atención a cómo, en la línea de acciones, va variando el nombre del objeto según cuánto lo hayamos observado.

```
PROCESS objeto(x, y, file);
PRIVATE
    // Hubo_seleccion nos indica
con TRUE que el bot4n de selecci4n
estaba
    //pulsado y con false que no
lo estaba.
    Hubo_seleccion = FALSE;
    // Hubo_colision nos indica
con TRUE que el cursor marcaba este
objeto en
    //la anterior evaluaci4n
    Hubo_colision = FALSE;
    Otro_obj = o_none;
    n = 0;
BEGIN
    // Inicializa variables
    NumID                      =
id_diana;
    Graph                      = 2;
    Estado                     =
to_diana;
    Nombre                     = "
cosa";
    EnlacesPrimarios[o_Ir_a]   = "
hacia ";
    EnlacesPrimarios[o_Hablar] =
" a";
    Ejecuta                    = FALSE;

    // Bucle principal del objeto
    LOOP
      // Movimiento del objeto de
prueba
      IF (Key(_o)) x-=10; END;
      IF (Key(_p)) x+=10; END;
      IF (Key(_q)) y-=10; END;
      IF (Key(_a)) y+=10; END;

      IF (COLLISION (TYPE
mouse))
        Hubo_colision = TRUE;
        IF (mouse.right)
          sel_accion(o_Mirar,
FALSE);          // Acción por
defecto
          mouse.left=TRUE;
// Ejecuta automáticamente
        END;
        IF ((mouse.left) &&
(!Hubo_seleccion))
          Hubo_seleccion = TRUE;
          Ejecuta = TRUE;
        ELSE
          IF (!mouse.left)
            Hubo_seleccion = FALSE;
          END;
          sel_objeto(id, FALSE);
        END;
      ELSE
        IF (Hubo_colision)
          Hubo_colision = FALSE;
          IF (Accion.id_obj_secun-
dario = id)
            Accion.id_obj_secunda-
rio = o_none;
          END;
        END;
      END;

      IF (Ejecuta)
        SWITCH (Accion.id_accion)
        CASE o_none:   sel_obje-
to(id, TRUE); END;
          CASE o_Ir_a:
acciones(o_none); END;
          CASE o_Hablar: accio-
nes(o_none); END;
          CASE o_Mover:  accio-
nes(o_none); END;
          CASE o_Mirar:
            SWITCH (n)
              CASE 0:
habla("Parece una diana "); Nombre
= ' posible diana'; END;
              CASE 1:
habla(")No crees que pueda ser una
diana?"); Nombre = ' dudosa diana';
END;
              CASE 2:
habla("Seguro que es una diana ");
Nombre = ' segura diana'; END;
              CASE 3:
habla("Te juro que es una diana de
ejemplo"); Nombre = ' diana de
ejemplo'; END;
              CASE 4:
habla("olvídame, no me echas cuen-
ta"); END;
              DEFAULT:
acciones(o_none); END;
            END;
            n ++;
            des_objeto();
          END;
          CASE o_Coger:  accio-
nes(o_none); END;
          CASE o_Dar:
acciones(o_none); END;
          CASE o_Abrir:
acciones(o_none); END;
          CASE o_Cerrar:
acciones(o_none); END;
          CASE o_Usar:
acciones(o_none); END;
        END;
        Ejecuta = False;
      END;
      FRAME;
    END;
END
Proceso objeto
```


**Indiana Jones, una invasión del mundo del cine que ha creado escuala.**

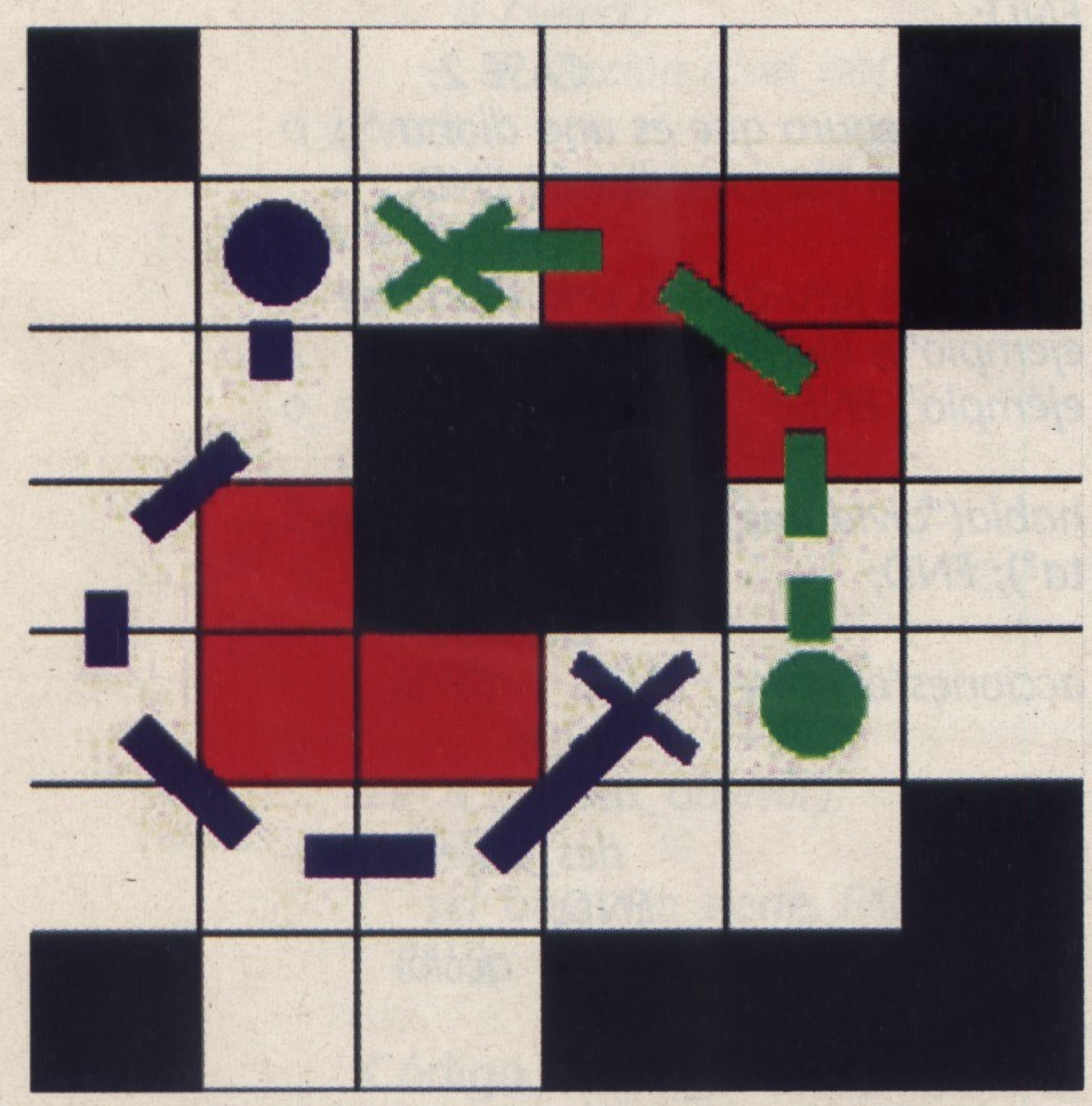

**Las tres dimensiones han llegado también a los grandes clásicos de la aventura.**

En el próximo artículo veremos cómo combinar acciones entre dos objetos (usar a con b) y crearemos el "algoritmo bolsa" con el que le permitiremos al jugador acarrear objetos por los distintos escenarios de nuestro juego.

**Miguel A. Barroso**

# Programación de Juegos de Estrategia (III)

**Tras dos números de iniciación en los que hemos empezado a pensar qué forma darle al juego, llega el momento de empezar a darle cuerpo. En este número tendremos oportunidad de ver y explicar el funcionamiento de algunos procesos necesarios para la realización de nuestro programa.**



**Caminos mínimos para llegar desde una bola a la cruz de su color**

Lo primero que vamos a comentar es la modificación de la estructura del proceso principal, que permite ampliar el numero de estados en los que puede estar el juego de forma rápida y sencilla. En el cuadro 1 se puede ver la nueva estructura que permite inicializar cada estado independientemente, mantener un bucle mientras se permanece en ese estado y pasar luego a otro tan solo cambiando dos variables y haciendo un *break*. Además, se ha añadido la posibilidad de parar el programa; tan solo con pulsar la P, se cargan los datos de fichero, se pinta el mapa según una matriz y se producen otros cambios que iremos viendo a continuación.

La rejilla de juego sirve para indicar por dónde se puede pasar y quién

## La función de pausa

Esta función (ver Cuadro 2) simplemente mantiene la variable en_pausa actualizada, según se pulse o no la tecla P. Cuando se está en pausa, pinta un gráfico a toda pantalla que debe tapar lo que hubiera (incluido el ratón), por lo que la z del proceso debe ser la mas pequeña de todas. La función continuamente comprueba si se ha pulsado la P, pero para que no ocurra el conocido efecto de múltiple captura de tecla, se hace un simple *FRAME* (1000), lo que evita que se compruebe la pulsación de la tecla hasta dentro de 10 *frames* (entre medio segundo y uno, según los fps). Como habréis podido comprobar, esta función no duerme ni hace nada parecido respecto al resto de procesos, sólo activa o desactiva un *flag*. Para que se dé el efecto pausa, habrá que cambiar todas las sentencias *FRAME* del resto de funciones por:

*repeat FRAME; until (not en_pausa);*

Una vez visto un método de hacer pausa (hay otros, por ejemplo usar las señales de aunque obligan a una fuerte estructuración de las llamadas a procesos), veamos como estructurar los mapas para que se pueda usar el tileado y los algoritmos de caminos basados en matrices.

### Cuadro 1

**Esqueleto del programa principal**

```
BEGIN
    Inicializaciones comunes a todos los estados.
    LOOP
        Parte común del bucle y acciones entre estado y estado (fade_off, etc.)
        if (Estado_del_programa == MENU && !cambio de estado)
            Inicialización del estado
            While (Estado_del_programa == MENU)
                Instrucciones correspondientes a este estado.
            end
        end
    END
END
```

## Tileado y rejillas

Lo mas sencillo es tener dos rejillas (matrices) sobre las que pueda transcurrir la acción del juego. Estas rejillas se representarán como una cadena de datos

## Cuadro 2

**Código de la función de control de pausa**

```
    PROCESS
Controlador_de_pausa()
    BEGIN
      graph=0;
      file=f_menus;
      flags=0;
      z=-1000000;

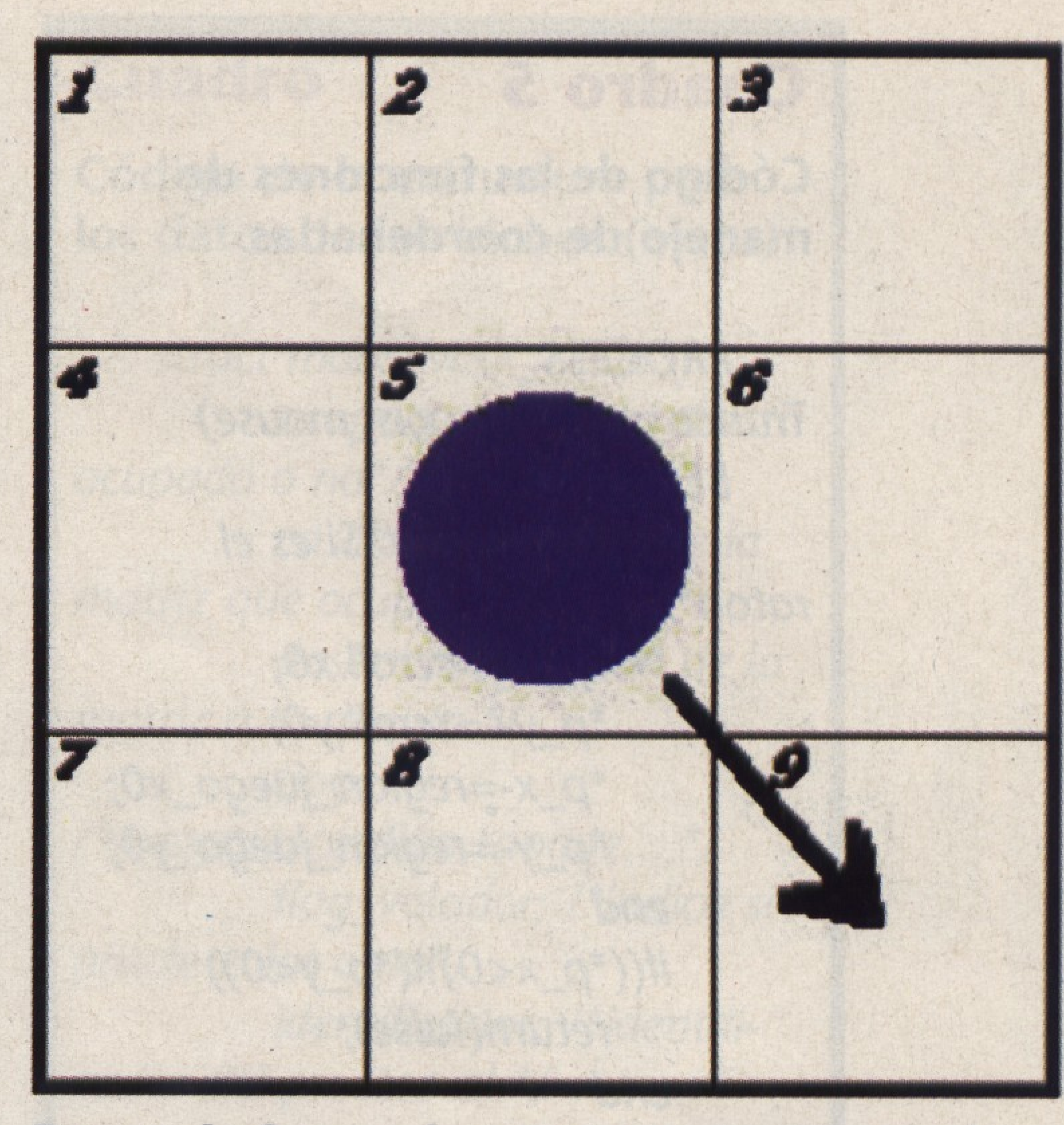
      loop
        if
(key(_P)&&(en_pausa==0))
          en_pausa=1;
          //colocamos un grafico de
PAUSA
          graph=9;
          FRAME(1000);
        end
        if
(key(_P)&&(en_pausa==1))
          en_pausa=0;
          graph=0;
          FRAME(1000);
        end
        FRAME;
      end
    END
```

## Cuadro 3

**Código de la función para pintar Mapas**

```
    PROCESS Pinta_Mapa(file)
    PRIVATE
      i_x,i_y;
    BEGIN
      x=y=0;
      graph=1;
      /*MAX_XMAP y
MAX_YMAP son el numero de tiles
verticales (horizontales) del mapa
*/
      for
(i_x=0;i_x<MAX_XMAP;i_x++)
        for
(i_y=0;i_y<MAX_YMAP;i_y++)

map_put(file,graph,rejilla_del_ma
pa[i_x+i_y*MAX_XMAP],
                        (i_x *
X_TILE)+(X_TILE/2),
                        (i_y *
Y_TILE)+(Y_TILE/2) );
        end
      end
    END
```




**Esta es la forma de marcar un avance en la matriz de juego.**

## Cuadro 4

**Código de la función de movimiento del scroll**

```
    PROCESS
camara_scroll(scroll.x0,scroll.y0,sp
eed)
    BEGIN
      priority=25000;
      loop
        if(scroll.x0 <= 0)
          scroll.x0=0;
        else
          if(mouse.x < 3 ||
key(_left)) scroll.x0-=speed; end;
        end

        if(scroll.y0 <= 0)
          scroll.y0=0;
        else
          if(mouse.y < 3 ||
key(_up)) scroll.y0-=speed; end;
        end

        if(scroll.x0 >=
MAX_XMAP*X_TILE)

scroll.x0=MAX_XMAP*X_TILE;
        else
          if(mouse.x > 317 ||
key(_right)) scroll.x0+=speed;
end;
        end

        if(scroll.y0 >=
MAX_YMAP*Y_TILE)

scroll.y0=MAX_YMAP*Y_TILE;
        else
          if(mouse.y > 237 ||
key(_down)) scroll.y0+=speed;
end;
        end

        repeat
          FRAME;
        until(!en_pausa);
      end
    END
```

[ancho * alto] dado que el DIV no permite manejo de varias dimensiones (el DIV 2 ya permite el uso de matrices, pero en este curso seguiremos manteniendo la compatibilidad DIV hasta que se generalice el uso de la segunda versión).

Tendremos una rejilla de juego por la que se moverán los procesos y que servirá para la búsqueda de caminos, indicándonos qué casillas son accesibles y cuáles no lo son (se puede decir que es una matriz de durezas). También tendremos una rejilla de mapeado, que contendrá el número de *tile* que debe haber en esa posición. Al guardar un mapa, sólo sería necesario guardar la matriz de mapeado, pero éste tiene zonas que son accesibles y otras que no, así que se guardan ambas rejillas. La función de pintado del mapeado es simple (ver Cuadro 3), tan solo recorre toda la matriz de mapa y pinta el *tile* correspondiente en esa posición del mapa de trabajo (un gráfico que será el primero en el fpg de *tiles,* y que tendrá un tamaño suficiente para que quepa todo el mapa completo).

La otra rejilla sirve para indicar por dónde se puede pasar y quién. En la figura 1 se muestra un ejemplo donde las casillas coloreadas marcan posiciones de la matriz que no están vacías, pero, mientras que las de color negro no las puede atravesar nadie, las de color rojo pueden ser atravesadas por un *mob* volador (el punto verde). Un *mob* no volador (el punto azul) tendrá que rodear las casillas rojas como si fueran negras. En el programa, los colores de cada casilla no son más que números. Así, el 0 significa libre, el 1 Monte, el 2 agua, etc.

Como el mapa es mas grande que la pantalla, utilizamos una ventana de *scroll* que moveremos acercando el ratón a los laterales, o pulsando las teclas de dirección. Para controlar este movimiento, la función *camara_scroll* modifica las variables *scroll.x0 y scroll.y0* (ver Cuadro 4). Para que este sistema funcione, *el scroll* deberá inicializarse en modo no automático, ya que de otro modo el *scroll* intentaría seguir a un proceso (habría que decirle a cuál). El problema del auto es que si el proceso cámara se acer-

## Cuadro 5

**Código de las funciones de manejo de coordenadas**

```
PROCESS
Trunca_xy(p_x,p_y,is_mouse)
BEGIN
    if(is_mouse) /*Si es el
raton*/
        *p_x+=scroll.x0;
        *p_y+=scroll.y0;
        *p_x-=region_juego_x0;
        *p_y-=region_juego_y0;
    end
    if((*p_x<0)||(*p_y<0))
        return(false);
    end
    *p_x=((*p_x/X_TILE) *
X_TILE)+(X_TILE/2);
    *p_y=((*p_y/Y_TILE) *
Y_TILE)+(Y_TILE/2);
    return(true);
END

/******************************
****/

PROCESS
posicion_xy(m_x,m_y)
BEGIN

return(m_x+m_y*MAX_XMAP);
END

/******************************
****/

PROCESS
xy_reales(p_x,p_y,casilla)
BEGIN

*p_x=(casilla%MAX_XMAP);
    *p_y=(casilla/MAX_XMAP);
    *p_x=(*p_x *
X_TILE)+(X_TILE/2);
    *p_y=(*p_y *
Y_TILE)+(Y_TILE/2);
    return;
END
```

## Cuadro 6

**Código de la función de manejo de bichos (*mobs*)**

```
PROCESS mob(file,num_id)
PRIVATE
    /* Sólo yo necesito estos
datos.            */
    /* Ademas no varian, asi
que no hay que salvarlos */
    camino[SEARCH_BUFFER];
    max_speed;
    speed;
    vista;
    damg_min;
    damg_max;
    fire_rate;
    alcance;
    precio;
BEGIN
    ctype=c_scroll;
    mobs[num_id].identifica-
dor=id;
    graph=mobs[num_id].tipo;

xy_reales(&x,&y,mobs[num_id].casi
lla);

    /* INICIALIZACION DE
DATOS CONSTANTES */
    switch(mobs[num_id].tipo)
        case TANK:

mobs[num_id].vida=TANK_VIDA;

max_speed=TANK_VELOCIDAD;

armadura=TANK_ARMADURA;
            vista=TANK_VISTA;

damg_min=TANK_DAMG_MIN;

damg_max=TANK_DAMG_MAX;

fire_rate=TANK_FIRE_RATE;

alcance=TANK_FIRE_LENGHT;
            precio=TANK_PRECIO;
        size=90;
        end
        case MOSCA:
        ...
        end
        .
        .
        .
    end
    LOOP

if(mobs[num_id].flag_volador)

rejilla_de_juego[mobs[num_id].casi
lla]=101;
        else

rejilla_de_juego[mobs[num_id].casi
lla]=100;
        end
        //si debo ir a algun sitio...
        if(mobs[num_id].desti-
no!=-1)
            //Esta función la vere-
mos el próximo número

//busca_camino(mobs[num_id].pos
icion,mobs[num_id].destino);

graph=mobs[num_id].tipo+1;
            FRAME(300);

graph=mobs[num_id].tipo+2;
            FRAME(200);
        end
        //si estoy muerto... me
salgo, y pinto una bonita explosión
:)...
        if(mobs[num_id].vida<0)

explosión(x,y,size,f_humo,10,9);
            mobs[num_id].ocupa-
do=0;
            return;
        end

        repeat
            FRAME;
        until(!en_pausa);
    END
END
```

ca hasta una esquina (el *scroll* no lo seguiría para no pintar vacío), al volver hacia el centro, el *scroll* tardaría un rato en reaccionar (lo cual queda bastante feo).

Para controlar el movimiento del *scroll*, la función *camara_scroll* modifica las variables *scroll.x0 y scroll.y0*

### Funciones de manejo de coordenadas

Para poder manejar las matrices de forma cómoda, se han implementado una serie de funciones (ver Cuadro 5) que pasan las coordenadas reales a posiciones de la rejilla y viceversa, así como una opción que modifica las coordenadas que le pasan, guardando en éstas los valores de x e y correspondientes al centro de rejilla mas cercano a ellas.

En el programa de ejemplo que viene en el CD-ROM, se puede observar algún ejemplo de su uso, como puede ser cuando, al pulsar en cualquier zona del mapa (tras seleccionar nuevo juego), aparezca una explosión centrada en la casilla sobre la que se pulsó.

### La estructura que guarda los datos de los *mobs* (bichos)

Dado que en DIV (V.1) sólo puede grabarse una vez por fichero, en consecuencia, sólo datos contiguos, entonces los datos personales de cada *mob* (sus variables locales), no pueden guardarse (salvo que guardes los datos de cada *mob* en un fichero



Pantalla del entorno de juego

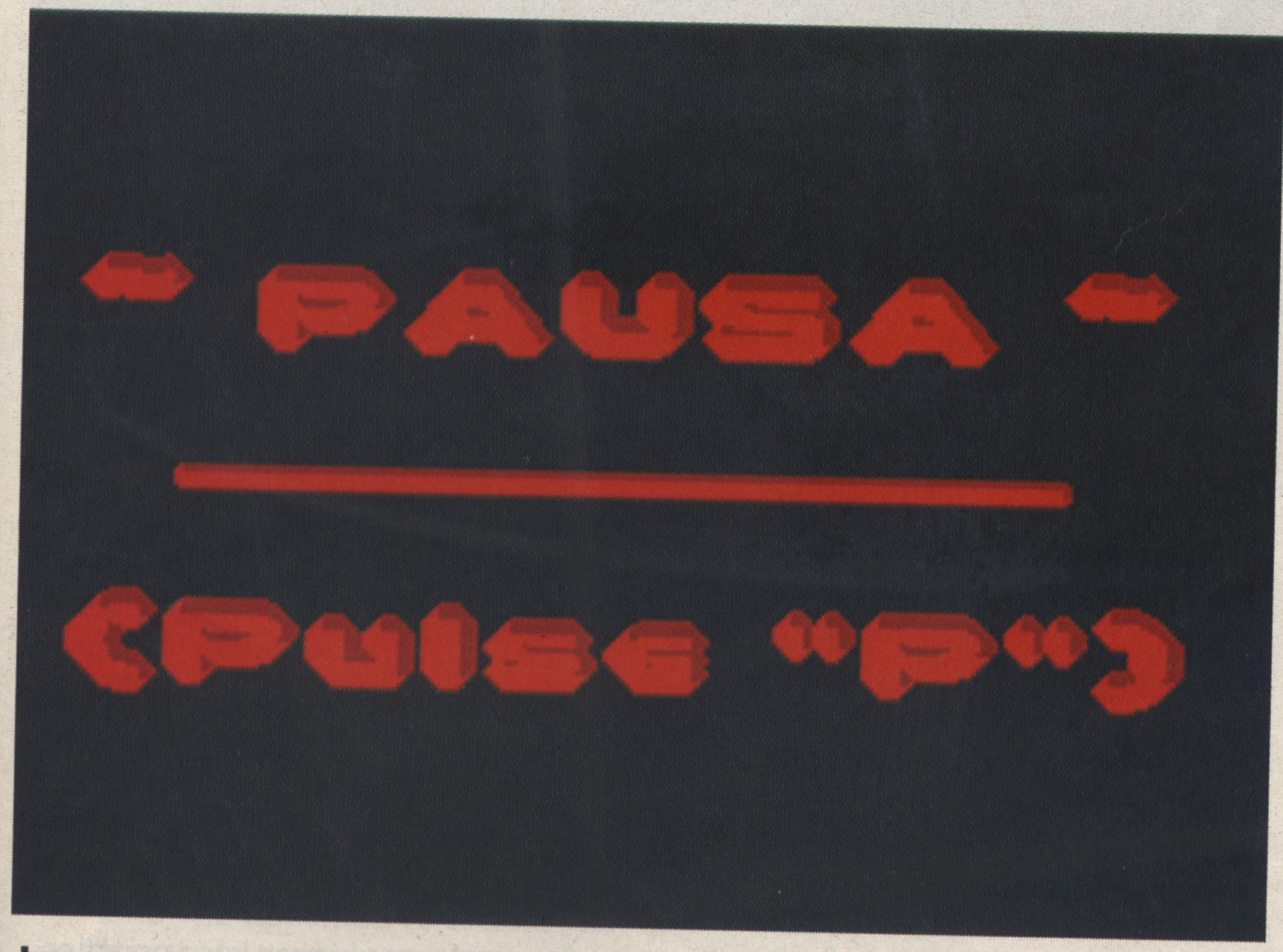



Imagen que aparece cuando pulsamos la P durante el juego.

distinto. No hará falta que comente que en un juego de estrategia el número de *mobs* llega a tener tres cifras). Para poder guardar el estado de la partida en cualquier momento, se mantienen en variables locales sólo los datos constantes (ya que se pueden recalcular en cada carga) y se guardan en una estructura los datos variables de cada *mob* (ver Cuadro 7). Como variable local también estará la posición de la estructura en la que se guardan los datos del *mob*. En la estructura, aparte de los datos para el juego, hay un *flag* de control (ocupado), que indica si en esa posición hay datos pertenecientes a algún *mob*, y una variable que guarda el identificador del *mob*, el cual guarda sus datos en esa posición (identificador). Conviene guardar también el numero de posiciones ocupadas para futuros usos. En sucesivos números de la revista ampliaremos el proceso *mob* ya que es el que controla todas las acciones de cada bicho, por lo que iremos viendo formas de gestionar las posiciones de la estructura.

## La función *mob*

A mas de uno le intrigará el porqué de llamar *mobs* a los bichos. *Mob* viene de la programación de *MUDS*, donde a los personajes no jugadores se les llamaba *mobs*. En un juego de estrategia todos los personajes son no jugadores (algunos cumplen ordenes del jugador, pero no son controlados enteramente por éste), así que parecía buena idea llamar así a la función que los controlaría. Esta función (ver Cuadro 6) se divide en dos partes: la inicialización de variables y constantes, y el bucle de juego (del cual sólo se sale o muriendo, en el juego, o "siendo matado", el proceso. Todo lo que puede hacer el *mob* (moverse, montar guardia, morir, etc.) debe estar considerado dentro de este bucle, y todos los datos que sean necesarios (para él o para otros) deben inicializarse antes de entrar en el bucle. No importa que luego nos acordemos de que nos hacía falta otro dato. Añadir un dato a la estructura es fácil, y por tanto puede hacerse cuando sea necesario.

La función *mob* se divide en dos partes: inicialización y bucle de juego

Para activar la pausa se debe pulsar la tecla P

## Y en el próximo número

En la próxima Divmanía veremos cómo salvar y cargar las partidas, así como la forma de buscar caminos en una matriz. Hasta pronto.

**Emilio Llamas ALba**

### Cuadro 7

**Código de la estructura para los datos de los bichos (*mobs*)**

```
struct mobs[MAX_MOBS]
        ocupado; /*marca si esta ocupado o no*/
        casilla; /*posición de la matriz que ocupa */
        destino; /*posición de la matriz a la que va */
        tipo; /*mosca,escorpión,tanque... */
        flag_volador; /*indica si puede volar o no*/
        identificador; /*identificador del proceso al */
                       /*que corresponden los datos*/
        bando;/*bando al que pertenecen*/
        vida; /*Obvio )no?*/
    end
    num_mobs; /* Numero de mobs almacenados */
```

# Cómo programar un juego de rol (II)

## Busca el elfo que llevas dentro

**En el anterior capítulo vimos los pasos que había que seguir antes de empezar a programar un juego de rol. En este segundo nos adentraremos en los sistemas de comercio en un juego de rol. Qué mejor manera para ello que analizando un ejemplo.**



Como ya comentamos en la anterior entrega, el sistema de comercio es el encargado de que tus personajes puedan comprar, vender o intercambiar objetos. El ejemplo que analizaremos es una simple demostración de un sistema de comercio basado en la compra/venta de objetos.

Ahora echémosle un vistazo al código del ejemplo:

```
Program comercio;
  Global
    fpg1;
    menun=3;
    menupos=1;
    comprarn=3;
    comprarpos=1;
    vendern=3;
    venderpos=1;
    seguros=2;
    dinero=1000;
    fuente;
    statust;
    espada=0;
    espadac=300;
    escudo=0;
    escudoc=200;
    casco=0;
    cascoc=100;
    comprart1;
    comprart2;
    comprart3;
    vendert0;
    vendert1;
    vendert2;
    vendert3;
    vendert4;
    vendert5;
    preguntat;
  Begin

fpg1=load_fpg("maqui.fpg");

fuente=load_fnt("nuevofnt.fnt");
    empieza();
    set_fps(9,0);
  END
```

Aquí declaramos todas las variables globales que vamos a usar en el programa. Las variables que acaban en:

- T: son variables de textos.
- C: son las variables que marcan el precio de un objeto.
- N: indican el número de posiciones posibles dentro de un menú.
- POS: marcan la posición actual dentro de un menú.

Otras variables son dinero y seguros, que se encargan de controlar el dinero que tenemos y de si estamos seguros de comprar/vender algún objeto.

En esta porción del código también cargamos el fichero de gráficos (variable fpg1) y la fuente que vamos a usar (variable fuente).

Desde aquí llamamos al proceso empieza que se encargará de



iniciar el juego. También usamos la sentencia set_fps para controlar el número de fps *(frames per second)* de nuestro programa.

```
process empieza()
  Private
  Begin
    set_mode(m320x240);
put(fpg1,5,320/2,240/2);
    Loop
      if(key(_enter))
        cursor();
  signal(type empieza,s_kill);
      END
      frame;
    END
  END
```

En este proceso inicializamos el modo gráfico a 320x240 pixeles y ponemos el gráfico de fondo mediante la orden put.

Una vez apretada la tecla enter se llama a la función cursor y el proceso se destruye.

```
process cursor()
  Private
yini=55;
    disty=23;
    xini=50;
Begin
    graph=3;
    x=xini;
    y=yini;
    size=50;
    menupos=1;
write(fuente,60,50,0,"Comprar");
write(fuente,60,73,0,"Vender");
write(fuente,60,96,0,"Salir");
    statust = write_int(fuen-
te,0,230,0,&dinero);
    Loop
      if(key(_down))
        y=y+disty;
        menupos++;
if(menupos>menun)
          menupos=1;
          y=yini;
          frame;
        END
      END
      if(key(_up))
        y=y-disty;
        menupos--;
        if(menupos==0)
menupos=menun;
y=yini+(disty*(menun-1));
          frame;
        END
      END
      if(key(_a))
        if (menupos==1)
          comprar();
signal(type cursor,s_sleep);
        END
        if (menupos==2)
              vender();
signal(type cursor,s_sleep);
        END
        if (menupos==3)
exit("Gracias por probar!",1);
        END
      END
      Frame;
    END
  END
```

Este proceso tiene 3 variables adicionales, que son:

xini : Marca la posición x inicial del cursor.

yini : Marca la posición y inicial del cursor.

disty : Es la distancia entre posiciones.

Estas 3 variables son muy útiles a la hora de modificar el programa, ya que si queremos cambiar la distancia entre líneas o la posición (x, y) donde empieza el cursor a moverse, tan sólo tendríamos que modificarlas, ahorrándonos cambiar otras partes del código.

A continuación se escriben los textos del menú, que son: comprar, vender y salir.

Inmediatamente después, se inicia el bucle principal, que se encarga de desplazar el cursor por la pantalla. Este bucle utiliza la variable menun para saber cuántas posiciones hay, y la variable menupos para indicar la posición en la que se encuentra actualmente.

Cuando se pulsa la tecla A se llama al proceso que le corresponda y se duerme el proceso actual. Para saber a qué proceso hay que llamar se usa la variable global menupos, que es la encargada de indicar en qué punto se encuentra el cursor.

```
process comprar();
private
disty=23;
yini=55;
xini=190;
begin
    graph=3;
    x=xini;
    y=yini;
    size=50;
    comprart1 = write(fuen-
te,200,50,0,"Espada");
    comprart2 = write(fuen-
te,200,73,0,"Escudo");
    comprart3 = write(fuen-
te,200,96,0,"Casco");
loop
        if(key(_down))
          y=y+disty;
          comprarpos++;
if(comprarpos>comprarn)
          comprarpos=1;
            y=yini;
            frame;
          END
        END
        if(key(_up))
          y=y-disty;
          comprarpos--;
        if(comprarpos==0)
          comprarpos=3;
y=yini+(disty*(comprarn-1));
            frame;
          END
        END
        if(key(_b))
      if (comprarpos==1)
            frame;
          comprueba();
      signal(type comprar,s_sleep);
```

```
            END
         if (comprarpos==2)
               frame;
            comprueba();
      signal(type comprar,s_sleep);
            END
         if (comprarpos==3)
               frame;
            comprueba();
      signal(type comprar,s_sleep);
            END
         END
      if(key(_esc))
   signal(type cursor, s_wakeup);;
      comprarpos=1;
   signal(type comprar, s_kill);
      delete_text(comprart1);
      delete_text(comprart2);
      delete_text(comprart3);
         END
   frame;
   END
   END

   process vender();
   private
   disty=23;
   yini=55;
   xini=190;
   begin
      graph=3;
      x=xini;
      y=yini;
      size=50;
      vendert0 = write(fuen-
te,200,50,0,"espada");
      vendert1 = write(fuen-
te,200,73,0,"escudo");
      vendert2 = write(fuen-
te,200,96,0,"casco");
      vendert3 = write_int(fuen-
te,250,50,0,&espada);
      vendert4 = write_int(fuen-
te,250,73,0,&escudo);
      vendert5 = write_int(fuen-
te,250,96,0,&casco);
   loop
         if(key(_down))
            y=y+disty;
            venderpos++;

if(venderpos>vendern)
         venderpos=1;
            y=yini;
            frame;
         END
      END
         if(key(_up))
            y=y-disty;
            venderpos--;
         if(venderpos==0)
            venderpos=3;

y=yini+(disty*(vendern-1));
            frame;
         END
      END
         if(key(_b))
      if (comprarpos==1)
            frame;
         compruebav();
   signal(type vender,s_sleep);
            END
         if (comprarpos==2)
               frame;
            compruebav();
   signal(type vender,s_sleep);
            END
         if (comprarpos==3)
               frame;
         compruebav();
   signal(type vender,s_sleep);
            END
         END
      if(key(_esc))
   signal(type cursor,s_wakeup);
         venderpos=1;
      delete_text(vendert0);
      delete_text(vendert1);
      delete_text(vendert2);
      delete_text(vendert3);
      delete_text(vendert4);
      delete_text(vendert5);
   signal(type vender, s_kill);
         END
   frame;
   END
   END
```

Estos dos procesos son muy similares.

Las variables que declaramos en estos dos procesos son las mismas que en la función cursor.

Lo primero que hacen tanto la función vender como comprar es escribir en pantalla los textos necesarios. La función comprar lo hace mediante la orden *write;* en cambio, el proceso vender añade otros textos mediante write_int, que indican el número de objetos disponibles de ese tipo. El número de objetos de cada tipo se almacena en una variable cuyo nombre es el mismo que el del objeto.

El bucle principal de estos dos procesos es muy similar a la función cursor, es decir, moverán el cursor dependiendo de las variables vender y comprar.

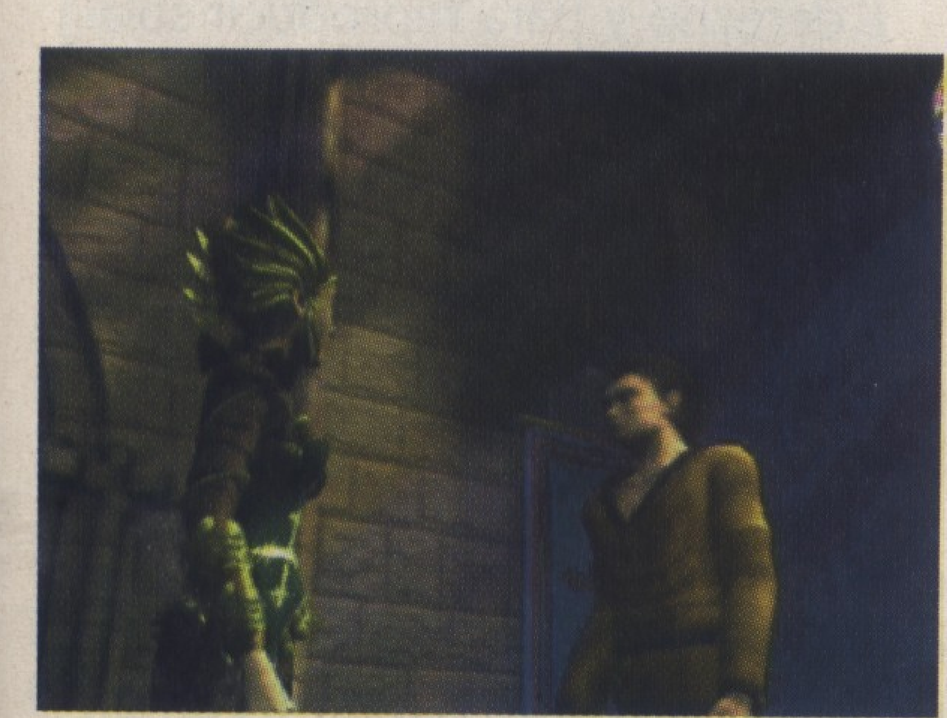
Cuando se aprieta la tecla B se consulta la variable comprarpos o venderpos, dependiendo si es en la función comprar o vender. Una vez comprobada la variable se duerme el proceso y se llama a comprueba (compruebav si es en la función vender). La función comprueba, que veremos más adelante, es la que se encarga de comprobar si se quiere y es posible comprar el objeto.

Si se pulsa la tecla ESC se borran los textos, se despierta a cursor, se pone la variable de posición de proceso a 1 (comprarpos y venderpos) y se destruye el proceso. Todo esto es necesario para hacer que se vuelva al menú.

```
process comprueba();
begin
preguntat =
write(fuente,0,0,0,"Está seguro que
desea comprar el objeto? (s/n)");
loop
   if (key(_s))
      seguros=1;
      comprueba2();
   END
   if (key(_n))
      seguros=0;
      comprueba2();
   END
   frame;
END
frame;
END
process comprueba2();
begin
   frame;
   delete_text(preguntat);
   signal(type comprueba,s_kill);
   if (seguros==1)
      if(comprarpos==1)
         dinero-=espadac;
      END
      if(comprarpos==2)
```

```
        dinero-=escudoc;
      END
      if(comprarpos==3)
        dinero-=cascoc;
      END
      seguros=2;
      if(dinero<0)
        statust = write(fuente,0,220,0,"No tienes suficiente dinero");
        if(comprarpos==1)
          dinero+=espadac;
        END
        if(comprarpos==2)
          dinero+=escudoc;
        END
        if(comprarpos==3)
          dinero+=cascoc;
        END
        signal(type comprar,s_wakeup);
        statust = write_int(fuente,0,230,0,&dinero);
      else
        statust = write(fuente,0,220,0,"Objeto comprado");
        signal(type comprar,s_wakeup);
        statust = write_int(fuente,0,230,0,&dinero);
        if(comprarpos==1)
          espada++;
        END
        if(comprarpos==2)
          escudo++;
        END
        if(comprarpos==3)
          casco++;
        END
      END
    END
    if (seguros==0)
      seguros=2;
      statust = write(fuente,0,220,0,"Objeto no comprado");
      signal(type comprar,s_wakeup);
      statust = write_int(fuente,0,230,0,&dinero);
    END
  END
```

Lo primero que hace el proceso es poner una pregunta en pantalla. El programa esperará a que se pulse la tecla S o la tecla N, que le indicará si queremos o no comprar el objeto. Si pulsamos S, la variable seguros se pondrá a 1, y si pulsamos la N se pondrá a 0. Después de darle un valor a seguros, se llama al proceso comprueba2.





El proceso comprueba2 funcionará en relación con el valor de la variable seguros. Lo primero que hará la función será matar al proceso que la llamó.

Mediante unas sentencias IF se seleccionará qué es lo que se va a hacer según el valor de seguros:

- Si es 0, se escribirá en pantalla un mensaje que indicará que no hemos comprado el objeto, se le asignará a seguros un valor diferente de 0 y 1, y se despertará a comprar.
- Si es 1, se comprueba la posición en la que estaba el proceso comprar (comprarpos), y se le resta a la variable dinero el valor del objeto (las variables acabadas en C son las que indican el coste de un objeto) que le corresponde a la posición donde se encontraba. Después se comprueba el valor de dinero: si es menor que 0, se escribe en la pantalla que no se puede comprar el objeto y se le devuelve el dinero que tenía antes; si es mayor que 0 se indica mediante un texto que se ha comprado el objeto y se incrementa en 1 la variable del objeto (espada, casco o escudo).

```
process compruebav();
begin
  preguntat = write(fuente,0,0,0,"Está seguro que desea vender el objeto? (s/n)");
  loop
    if (key(_s))
      seguros=1;
      compruebav2();
    END
    if (key(_n))
      seguros=0;
      compruebav2();
    END
    frame;
  END
  frame;
END
process compruebav2();
begin
  frame;
  delete_text(preguntat);
  signal(type compruebav,s_kill);
  if (seguros==1)
    seguros=2;
    if(venderpos==1)
      espada--;
    END
    if(venderpos==2)
      escudo--;
    END
    if(venderpos==3)
      casco--;
    END
    if( espada<0 or escudo<0 or casco<0 )
      if(venderpos==1)
        espada++;
      END
      if(venderpos==2)
        escudo++;
      END
      if(venderpos==3)
        casco++;
      END
      statust = write(fuente,0,220,0,"No tienes ese objeto");
      signal(type vender, s_wakeup);
      statust = write_int(fuente,0,230,0,&dinero);
    else
      statust = write(fuente,0,220,0,"Objeto vendido");
      signal(type vender, s_wakeup);
      if(venderpos==1)
        dinero+=espadac;
      END
      if(venderpos==2)
        dinero+=escudoc;
      END
      if(venderpos==3)
        dinero+=cascoc;
      END
      statust = write_int(fuente,0,230,0,&dinero);
    END
  END
  if (seguros==0)
    seguros=2;
    statust = write(fuente,0,220,0,"Objeto no vENDido");
    signal(type vender,s_wakeup);
    statust = write_int(fuente,0,230,0,&dinero);
  END
END
```

Estas dos funciones trabajan de una manera muy similar a las anteriores, así que sólo comentaré los cambios que se encuentran en compruebav2.

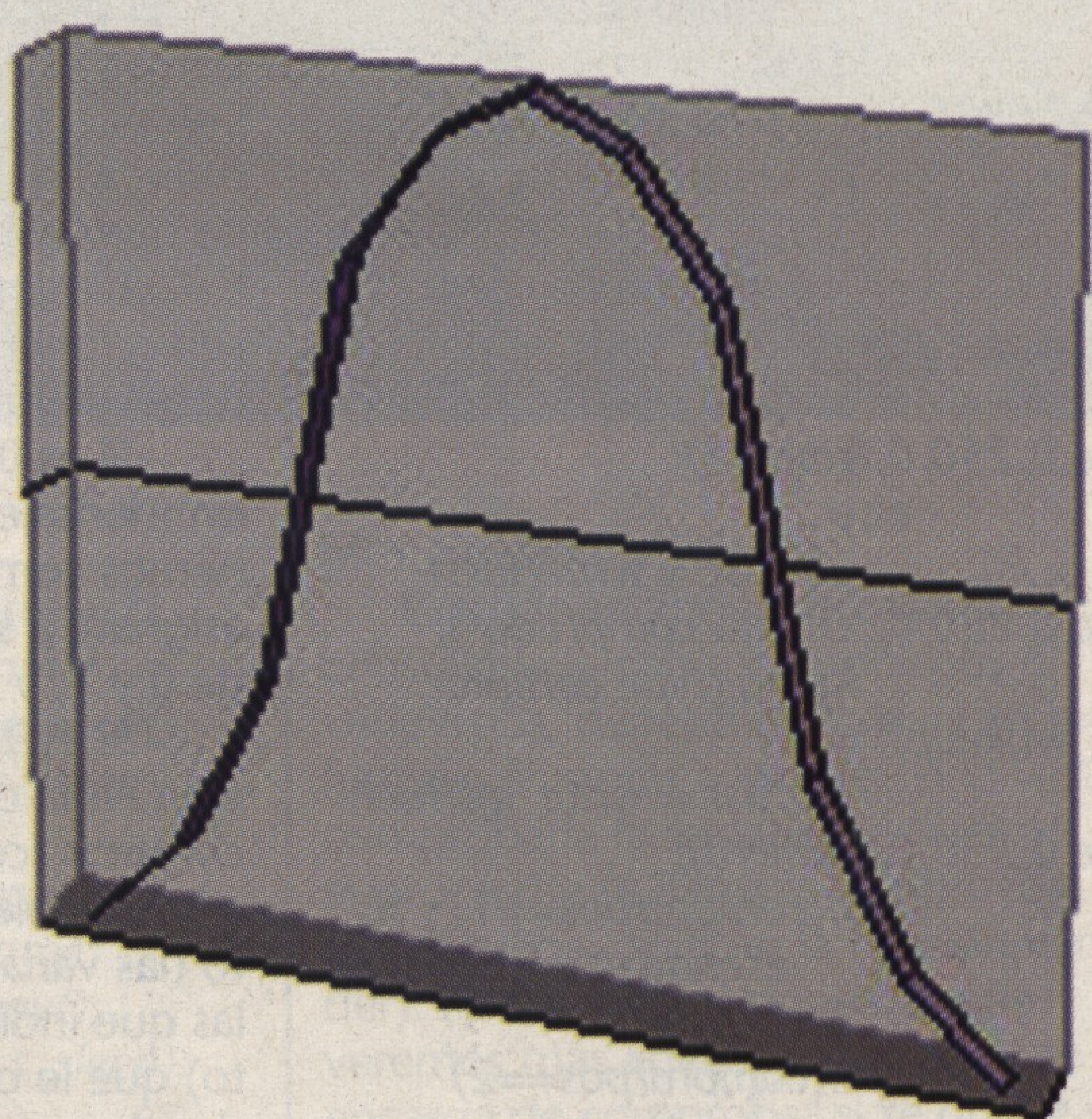
Lo primero que hace compruebav2 es restarle 1 a la variable de objeto según su posición. Después se mira si hay algún objeto cuya variable de objeto es menor que 0; si es así se le suma 1 y se indica que no se puede vender el objeto. Si no hay ninguna variable menor que 0 se imprime en pantalla un mensaje indicando que se ha comprado el objeto y se suma a dinero el valor del objeto vendido.

**Ramón de España Oliu (MAQNAZILLER)**
**respana@arrakis.es**

# Fasttracker 2.08 2ª parte

## Haciéndonos maestros de nuestra herramienta musical

**Una vez más, nos adentramos en el mundo de la creación de sonido por ordenador con el fin de aplicar los conocimientos en nuestros propios juegos. En el número anterior nos introdujimos de manera práctica, aunque poco rigurosa, en el manejo del programa Fasttracker 2.08. En esta ocasión prestaremos una mayor atención al manejo del programa, dejando un poco apartados los temas puramente musicales.**



**Ejemplo de envolvente de volumen *(volume envelope).***

A estas alturas ya deberíamos estar familiarizados con el manejo de Fasttracker, como herramienta para la composición y ejecución de temas musicales, e incluso ser capaces de crear composiciones sencillas. Siendo así, estamos listos para profundizar en los vericuetos del programa, para dar a nuestras canciones ese toque que necesitan, y para ambientar de manera más efectiva nuestros juegos.

### Cosas importantes acerca de Fasttracker 2.08

Algo que va a resultar imprescindible para que lleguemos a dominar el programa es sentirnos cómodos con él. Si utilizamos el programa bajo Windows, seguramente no experimentaremos esa sensación de comodidad, ya que sufriremos retardos. Esto es algo que desmerecerá considerablemente el sonido de nuestras creaciones musicales, haciendo que éstas vayan a trompicones y que se hagan insoportables cuando intentemos realizar acciones tan usuales como digitalizar un sonido desde el editor de *samples*. Por ello recomendamos ejecutar el programa bajo MS-DOS.

Es importante llegar a sentirnos cómodos con el programa para poder dar rienda suelta a nuestra creatividad

Así mismo, es importante destacar que Fasttracker suele tener problemas con la detección de conocidas tarjetas de sonido, como la SoundBlaster 64 Awe y los modelos PCI de la misma marca, ya que no puede ser configurado para ciertas IRQ=s en las cuales tienen costumbre de alojarse estas tarjetas. Una solución del problema es configurar en Windows la tarjeta de sonido para que comparta un IRQ que sí pueda detectar Fasttracker, con otro dispositivo (la impresora suele ser una buena solución), reiniciando en modo MS-DOS posteriormente.

Otro detalle significativo es que para poder hacer uso de nuestras canciones junto con la herramienta DIV de creación de juegos, tendremos que grabarlas en formato WAV, que es el único formato compatible con Fasttracker, que hasta el momento soporta el DIV. Para conseguir la última versión de Fasttracker: http://www.starbreeze.com

### Dominando las opciones del teclado

Fasttracker es un programa bastante complejo que comprende gran cantidad de acciones, de manera que casi todas las teclas de nuestro teclado tendrán asignada una función determinada. Es importante que tengamos cierta soltura en el manejo de estas teclas para una mayor rapidez en el uso del programa. Algunas útiles son:

- Activación / desactivación de la opción de escritura: Barra espaciadora.
- Escribir un silencio en una pista: <Bloqueo Mayúsculas>.
- Selección de octava: con las teclas de función de <F1> a <F7>, podremos seleccionar la octava en la que deseemos reproducir cada instrumento, obteniendo con <F1> el tono más grave y con <F7> el tono más agudo.
- Movimiento del cursor en la partitura: <TAB> para saltar entre pistas y <F9>-<F12>, <Re. Pág>, <Av. Pág>, <Incio> y <Fin> para desplazamientos instantáneos en vertical.
- Para reproducir una canción haremos uso del <Control> derecho y para reproducir solamente un pattern: <Alt Gr>.
- Grabar un pattern: <Shift> derecho.
- Por último, para borrar una nota usaremos <Del> y para insertarla <Ins>.

Algo que resulta de extrema utilidad es controlar las funciones de copiar, cortar y pegar, que nos evitarán tener que repetir fragmen-

tos manualmente. En este campo, todas las acciones referentes a pistas *(tracks)* se harán pulsando la tecla <Shift>, las de *patterns* con <Control> y las referentes a selecciones hechas por el usuario con <Alt>. Habrá que pulsar en cada caso, junto con las citadas anteriormente <F3> para cortar, <F4> para copiar y <F5> para pegar. Así, por ejemplo, la combinación <Alt+F4> copiaría un bloque de notas seleccionado por el usuario con el ratón y <Alt+F5> pegaría el bloque copiado en la posición actual del cursor en la partitura.

Por supuesto, son muchas más las teclas (y combinaciones de teclas) que ejecutan acciones, pero, en principio, será suficiente con las que hemos comentado. En caso de querer profundizar aún más en las funciones del teclado, podremos remitirnos a la opción *Help,* y posteriormente a la sección *Keyboard* de la misma.

## Edición de *samples*

Fasttracker es, además de un excelente secuenciador, un buen editor de *samples.* Podemos utilizar esta herramienta del programa cuando queramos importar algún sonido de una fuente externa, sin necesidad de hacer uso de ningún otro programa de edición extra, como por ejemplo, Cool Edit 96 (del cual tratamos en el primer número de la revista).

Pulsamos en la opción Smp. Ed y observamos que el editor que se nos presenta es muy intuitivo, y nos ofrece la posibilidad de elegir diferentes factores, como la frecuencia de muestreo del sonido (a mayor frecuencia, mayor calidad de reproducción), el formato de sampleo (estéreo o mono) y el número de bits que utilizaremos para representar cada sonido (8 ó 16). Una

**Ejemplo de envolvente de panorámica (panning envelope).**

### Efectos en tiempo real

| Efecto | Código | Parámetros (x,y) |
|---|---|---|
| Slide de tono arriba | 1 | velocidad (dos dígitos) |
| Slide de tono abajo | 2 | velocidad (dos dígitos) |
| Vibrato | 4 | velocidad, profundidad |
| Volumen | C | volumen (dos dígitos) |
| Slide de volumen | A | vel. de subida, vel. de bajada |
| Selecciona canal | 8 | posición (dos dígitos) |
| Salto de pattern | B | número del pattern (dos dígitos) |
| Saltar a siguiente pattern | D | línea de inicio (dos dígitos) |

opción muy interesante es la de elegir la forma de loop con la que queremos que se reproduzca nuestro sonido, es decir, si queremos que sólo se reproduzca una vez o si queremos que lo haga varias veces, y de qué manera queremos que se repita: volviendo de nuevo al principio del sample (modo *forward*), o reproduciéndolo en reversa para luego volver a repetirlo hacia delante (modo *pingpong*). Esta opción no se encuentra en otros editores de similares características y es muy importante para la realización de canciones para juegos, ya que, en principio, si queremos que se esté reproduciendo una canción mientras se desarrolla la acción del mismo, será deseable que esa canción no ocupe mucha memoria RAM para dejar espacio al resto de elementos del programa, tales como los gráficos. Podremos entonces digitalizar un fragmento corto de un sonido que, en teoría, deba durar más, y hacer que se repita infinitamente mediante un loop de tipo *pingpong*, consiguiendo limpiamente el efecto deseado y economizando una cantidad considerable de memoria (ya que 1 minuto de sonido en calidad CD, ocupa 10 Mb de RAM, lo cual es mucha RAM).

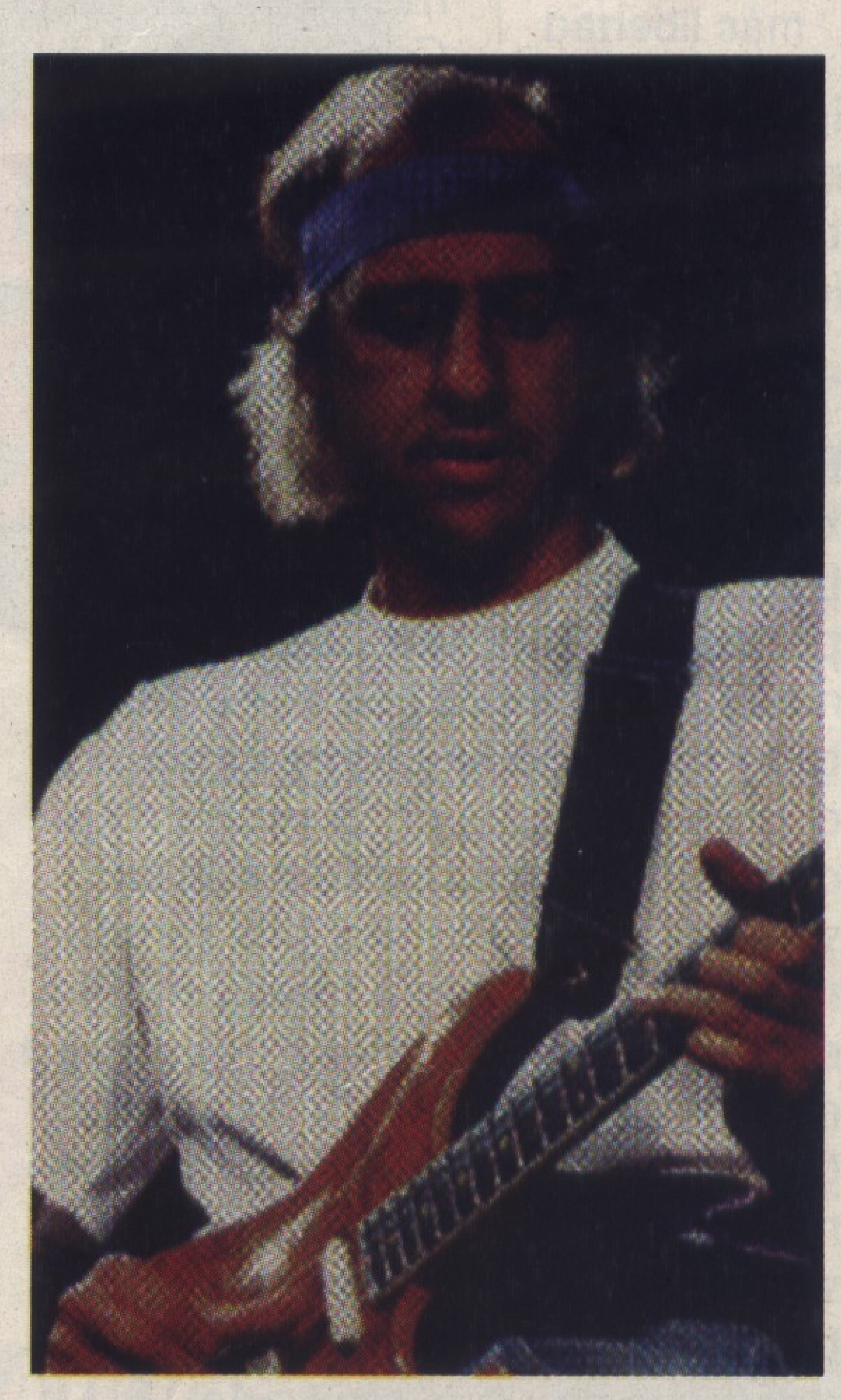
Además de lo citado, disponemos de las típicas opciones para cortar, pegar y copiar fragmentos de onda, así como la posibilidad de modificar el volumen de determinadas partes del sonido digitalizado.

## Edición de instrumentos

Una vez que tenemos la materia prima con que conseguir un nuevo instrumento para nuestra flamante composición musical, necesitamos terminar convenientemente nuestra labor. Para ello haremos uso del editor de instrumentos, al cual accedemos mediante el botón Instr. Ed. Aquí se nos da la opción de aplicar dos envolventes a nuestra onda, los cuales la modificarán a lo largo del tiempo que esté sonando: una que influirá en el volumen de la misma, y otra que influirá en el canal por el que emerge (izquierdo o derecho). Para hacer efectivo el uso de esta última, es importante tener en cuenta que debemos tener activada la función de reproducción en estéreo, que se encuentra en la parte derecha de la pantalla cuando accedemos al menú Config. Podremos modificar la forma de las envolventes libremente para conseguir el efecto sonoro deseado. Veamos unos ejemplos:

Gracias al editor de samples y al de instrumentos podremos generar nuestros propios sonidos con facilidad

Imaginemos que queremos incluir en uno de nuestros temas el sonido de una sirena. Para ello bastará con samplear la parte del sonido que se repite cíclicamente y luego aplicarle un loop de tipo forward. Ahora queremos que esa sirena empiece sonando como si estuviera muy lejos y que se vaya acercando paulatinamente para terminar alejándose otra vez. Este



**Con los efectos en tiempo real podemos emular de manera bastante efectiva la expresividad de una guitarra eléctrica.**

efecto no podríamos realizarlo con el editor de *samples,* a no ser que digitalizaramos una porción de sonido lo suficientemente grande como para aplicarle el efecto completo, lo cual supondría un enorme gasto de memoria. La forma correcta sería aplicarle al *sample* una envolvente de volumen (*volume envelope*) convexa, es decir, con forma de U invertida, que parta de cero, que crezca hasta el máximo, y que luego disminuya hasta volver a hacerse cero (fig.1).

Ahora, para hacer completo el efecto, queremos que nuestra sirena comience sonando desde el canal izquierdo y que acabe sonando exclusivamente por el derecho, para lograr así un destacable efecto de panorámica. Para conseguir esto, debemos tener en cuenta que la gráfica de la envolvente de panorámica (*panning envelope*), es una gráfica tiempo (eje x) B posición (eje y), de manera que un punto en la parte más inferior de la gráfica hará que el sonido se ubique en el canal derecho, mientras que si está en la parte superior logrará el efecto contrario, es decir, que el sonido se ubique en el canal izquierdo. De la misma forma, un punto en la parte central del eje y de la gráfica hará que el sonido salga con el mismo volumen por los dos canales. Así, para conseguir el efecto que queríamos para la sirena, tendremos que hacer una línea inclinada que comenzará en el extremo superior izquierdo de la gráfica y acabará en el extremo inferior derecho de la misma (fig.2).

Los numerosos efectos en tiempo real nos ayudarán a expresarnos con más libertad

Con el editor de instrumentos, también podremos modificar el tono de los instrumentos, en octavas y semitonos, e incluso en rangos muy pequeños.

## Transporte

El transporte es un concepto algo más complejo. Básicamente, se trata de disminuir o aumentar el periodo de tono de una o varias notas, de manera que el nuevo tono obtenido se corresponda con el de otras notas de la escala musical. Por ejemplo, transportar la secuencia La - Fa - Sol dos semitonos más abajo, sería transformarla en la secuencia Sol - Re - Fa, que a efectos prácticos nos sonaría prácticamente igual, pero, dados determinados casos, nos podría interesar realizar uno de estos transportes. Transportar es un proceso tedioso si hay que realizarlo manualmente sobre una secuencia de notas muy larga o compleja. Este proceso queda totalmente mecanizado con la herramienta *Transps.* de Fasttracker. Veamos una vez más un caso práctico:

Hemos realizado una estupenda composición musical, la cual quisiéramos completar con un fragmento de letra cantada por nosotros mismos. Hemos conseguido una línea melódica que nos satisface y nos disponemos a grabarla. Para ello, grabamos en una cinta de cassette nuestra canción para reproducirla mientras cantamos, sampleando nuestra voz. Cuando intentamos acoplar el *sample* realizado a nuestro módulo, observamos que, pese a que creemos que afinábamos cuando cantábamos, parece que los tonos no coinciden. Esto podría deberse a la diferente velocidad de reproducción del cassette, que afecta también al tono. Por regla general, estos desajustes suelen oscilar en el rango de un semitono. Así que para solucionar nuestro problema bastaría con acceder a la opción *Transps.,* y pinchar una vez con el ratón en el botón dn de la sección *Song* de la parte *All instruments,* transportando así un semitono hacia abajo toda nuestra canción, para amoldarla al fragmento cantado que queremos introducir.

Por supuesto, se podrán realizar transportes de fragmentos más pequeños (no sólo de la canción entera) tales como una pista, un *pattern,* o un bloque seleccionado por el usuario. También podremos elegir que el transporte sólo afecte a un instrumento determinado.

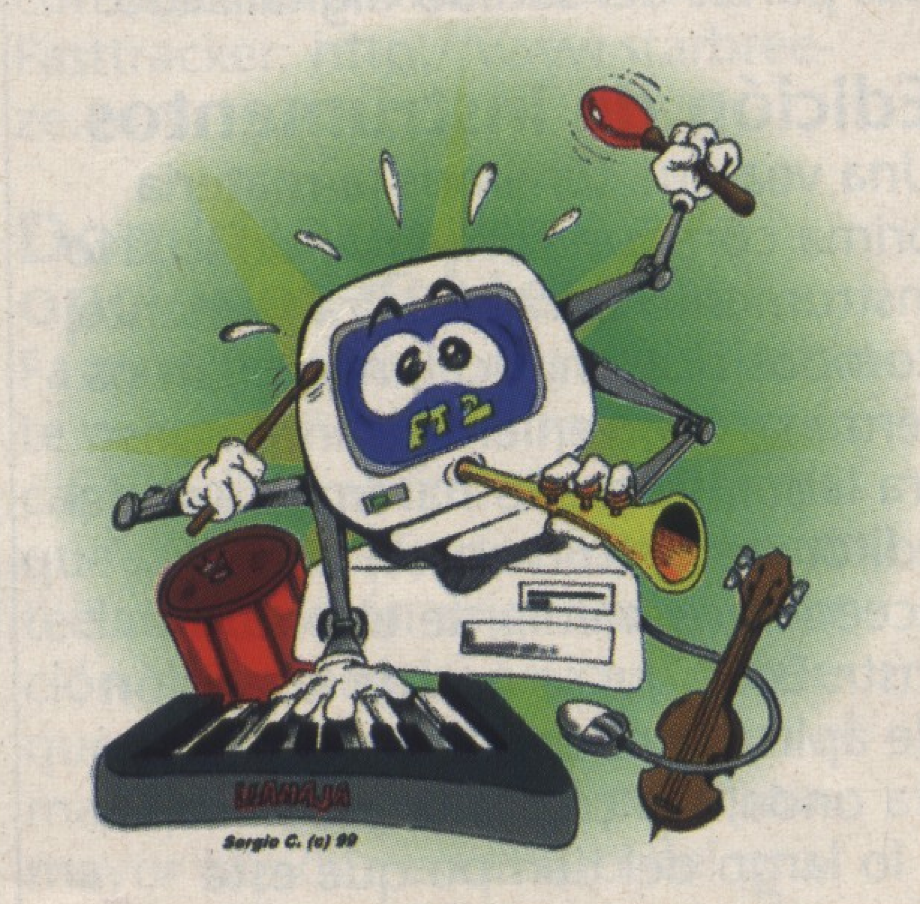
## Efectos en tiempo real

Ya mencionamos en el anterior artículo que existe la posibilidad de aplicar efectos en tiempo real a los sonidos, haciendo uso de los tres últimos dígitos de cada pista. De estos tres dígitos, el primero corresponde al código de identificación del efecto, y los dos siguientes a los parámetros del mismo. Dependiendo del tipo de efecto, tendremos que poner su código una o varias veces. Los efectos más interesantes son los siguientes:

- *Slide* de tono: entendemos por *slide* un cambio gradual (en este caso de tono). Pondremos 1 como código para que el *slide* sea hacia arriba, y 2 para que sea hacia abajo, y con los dos dígitos de parámetro indicaremos la velocidad del *slide.*
- *Vibrato:* produce una vibración en el sonido al que apliquemos el efecto. Se activa con el código 4 y los dígitos siguientes marcan, respectivamente, la velocidad y la profundidad de la vibración.
- Volumen: selecciona el volumen (de 0 a 40 Hexadecimal) en esa posición exacta. El código es C.
- *Slide* de volumen: provoca un cambio gradual en el volumen de reproducción del canal en cuestión. Se activa con el código A y sus parámetros indican la velocidad de subida o bajada de volumen, respectivamente, de manera que uno de los dos debe ser cero.
- Seleccionar posición en la panorámica: asigna la posición del sonido de la pista, es decir, si debe sonar más por la izquierda o por la derecha. Le corresponde el código 8 y los dos dígitos siguientes indican la posición exacta: 00 es la que estaría más a la izquierda, y FF la que estaría más a la derecha.
- Salto de posición: salta a otro *pattern.* El código es la B y su parámetro el *pattern* al que se desea saltar.
- Salto al siguiente *pattern* en la posición indicada: salta al *pattern* que siga en la lista reproduciéndolo desde la posición que se le introduzca como parámetro. Su código es la D.

Al igual que con la opciones de teclado, hay más efectos, pero los citados son los más utilizados. En caso de querer explorar más efectos, aconsejamos una vez más dirigirse a la ayuda, en la sección *Effects.*

Vistas ya opciones un poco más avanzadas, y con un buen número de horas de práctica, estaremos dispuestos a tratar, en próximas entregas, temas más complejos en el campo de composición musical basada en Fasttracker.



**Sergio Cánovas**

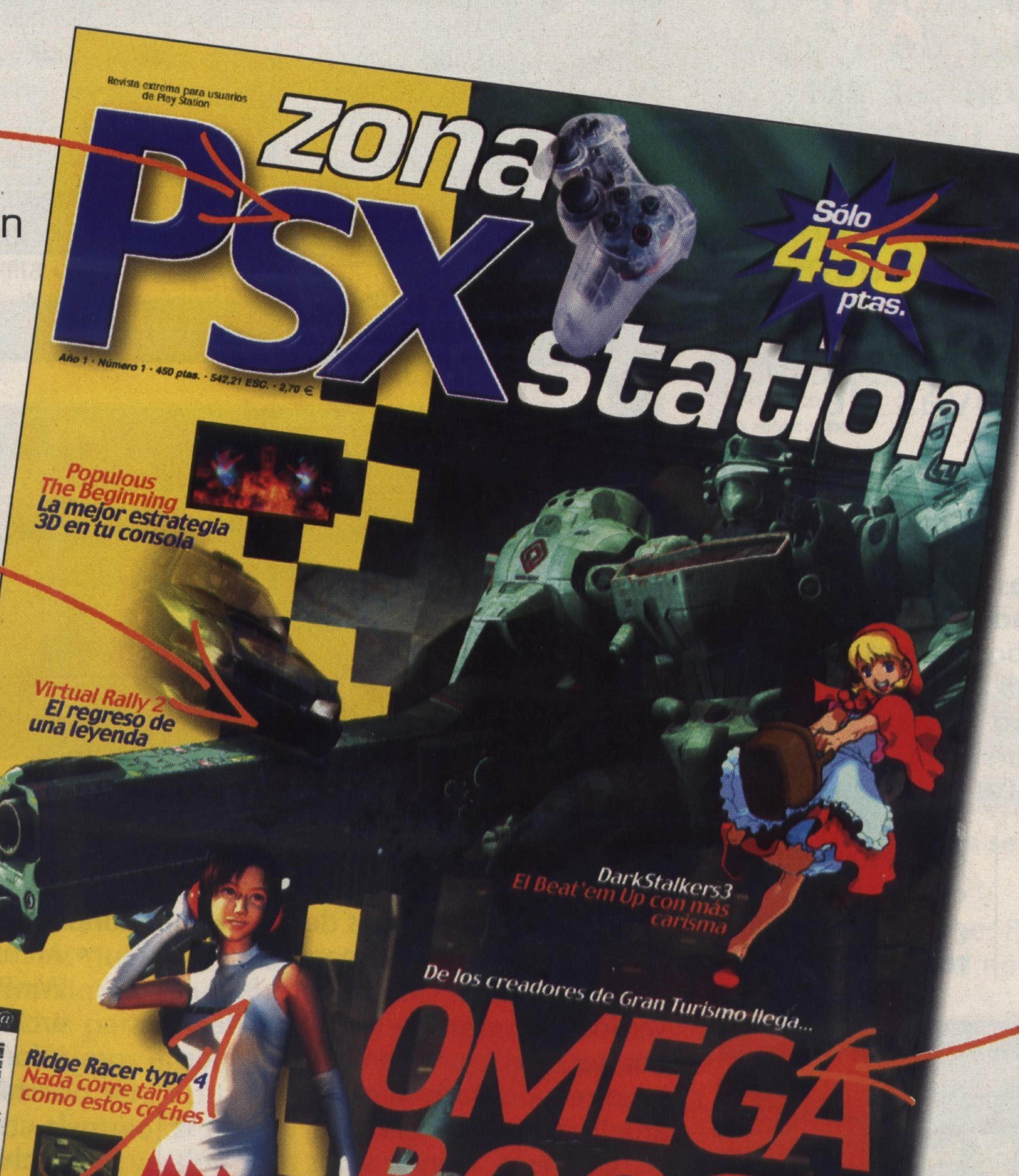
# Para que luego no digas que **en una** sola **revista** no cabe **todo** lo que buscas **sobre PlayStation**

Una revista **visual** con **comentarios amenos**. Para que la disfrutes **leyendo y hojeando.**

Todos los **juegos, noticias, utilidades, trucos, previews y reportajes** relacionados con PlayStation.

**sólo 450 Ptas.**
2,70€

Una **evolución** objetiva de los **juegos** más **importantes** que puedes encontrar en el **mercado.**




**Zona Manga, periféricos, otras consolas, ciencia-ficción...** porque, como tú, sabemos que **PlayStation** es mucho más que una consola.

En nuestra **sección Preview** te adelantamos los **títulos** que harán **furor** en la **próxima temporada.**

Con cada número, un **suplemento** de **16 páginas** dedicadas al **bombazo del mes.**

**No podían faltar as noticias.** La actualidad a tu disposición para que o te pierdas ni un detalle.

**Trucos, trucos y más trucos.** No habrá juego que se te resista.

## ¡YA DISPONIBLE!

A la **venta** en **quioscos, librerías, grandes almacenes** y **tiendas especializadas.**


Edita PRENSA TÉCNICA •
C/ Alfonso Gómez, 42 - Nave 1-1-2 •
28037 Madrid España (Spain) •
o: 91. 304.06.22• Fax: 91. 304.17.97





www.prensatecnica.com

**PRENSA TÉCNICA te ofrece**

## LA REVISTA DE PLAYSTATION MÁS AMENA Y RIGUROSA DEL MERCADO

# WinPatch 1.2

## Parches para programas

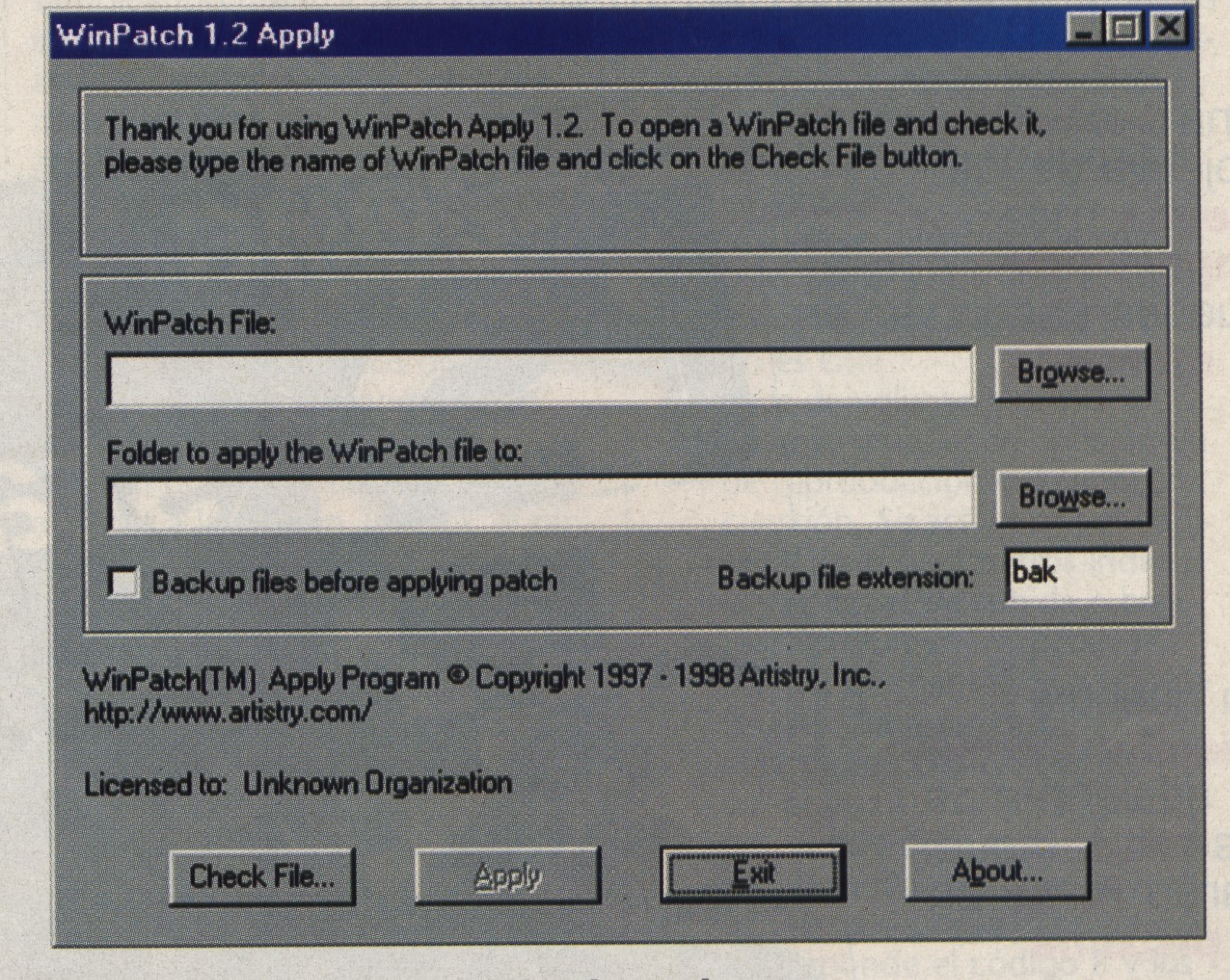
**En esta edición de la revista se va a hablar de WinPatch 1.2, en su versión de 32 bits (Windows 95/98/NT). Se trata de un programa que está básicamente orientado a hacer más fácil la actualización de paquetes de software, sobre todo a través de Internet.**

Cuántas veces nos hemos quejado de la cantidad de tiempo y dinero que nos cuesta actualizar la última versión nuestro software (ya sea *freeware* o *shareware*) a través de Internet? WinPatch está especialmente creado para estas situaciones. Pero ¿cómo funciona realmente?

Con la ayuda de WinPatch es posible la distribución de nuevas versiones de programas por Internet de forma mucho más cómoda




**Si lo hemos hecho todo correctamente, nos saldrá esto.**




**Aspecto del programa aplicador de parches.**

Para realizar un parche, es necesario tener dos versiones del mismo programa, a las que llamaremos antigua y nueva. Obviamente, la versión nueva es una versión posterior del mismo programa, ya sea por la corrección de *bugs* o por la ampliación del programa.

Con esto, le indicamos a WinPatch donde están las distintas versiones; el propio programa busca las diferencias entre ellas y crea un archivo, que aplicándoselo a la versión antigua, la convierte en la versión nueva.

### Instalación y configuración

La instalación del programa es muy sencilla. Viene en un paquete de distribución autoextraible (*Package for the webtm de InstallShield*). Una vez instalado vemos que existen dos formas de funcionamiento del programa: el modo *Expert* y el modo *Wizard,* así como un pequeño programa de aplicación de parches.

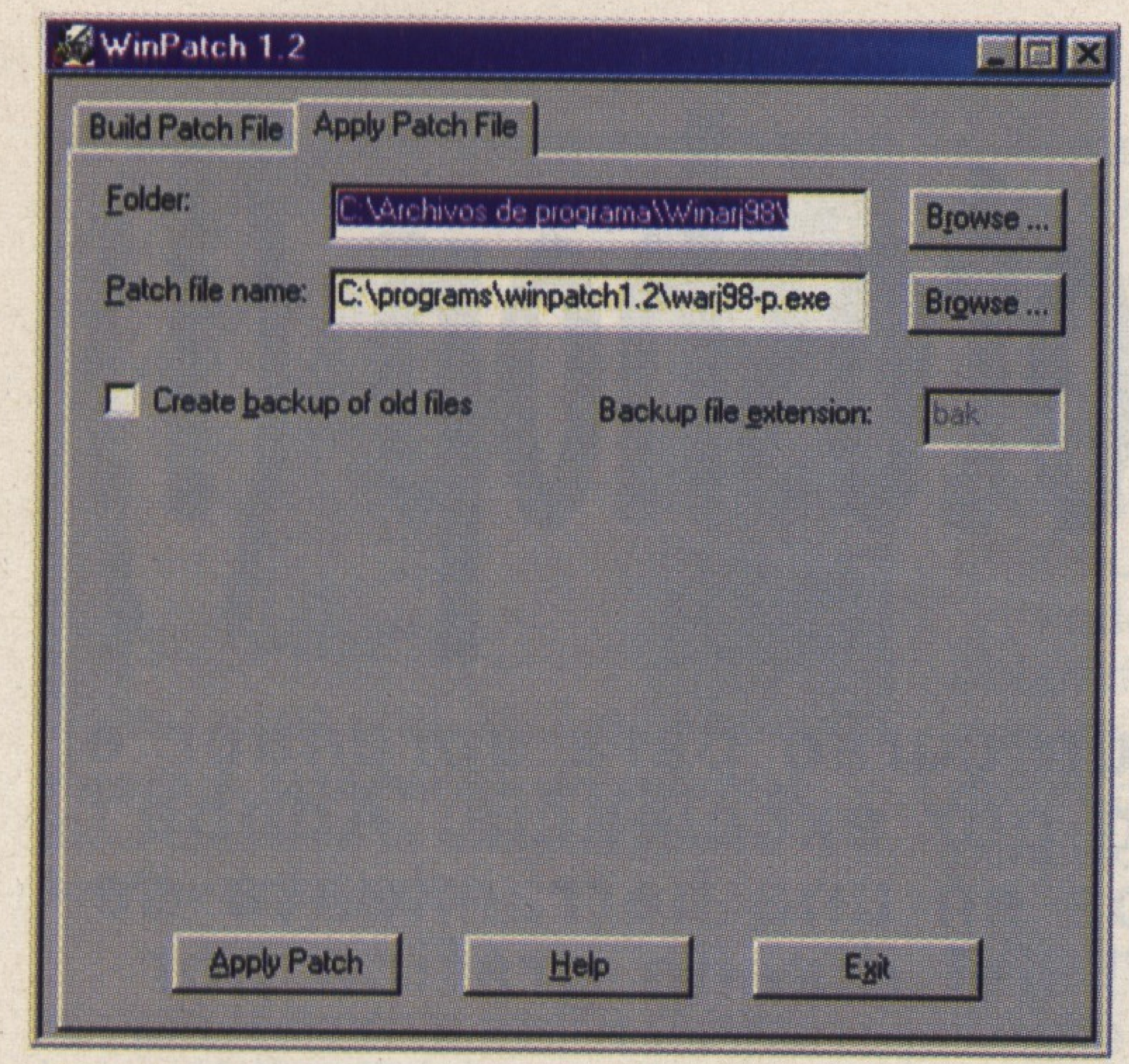
### Un breve repaso

Antes que nada, es necesario aclarar qué clase de archivos utiliza Winpatch:

- Archivos de proyecto de WinPatch (.wpp): en estos archivos se guarda toda la información necesaria, tanto archivos como opciones, para la creación de un parche.
- Archivos de parches de WinPatch (.wnp): estos archivos guardan toda la información necesaria para la actualización del programa en cuestión.
- Archivos de parches ejecutables de Winpatch (.exe): no son más que archivos .wnp a los cuales se les ha añadido un pequeño programa que permite la autoinstalación del parche.

### Crear un parche

Para la creación de parches podremos utilizar un proyecto ya creado o crear uno nuevo. Por ejemplo: estamos realizando una aplicación, la cual estamos distribuyendo por Internet. Corregimos y ampliamos el programa, y la nueva versión (pongamos 0.9b) la utilizamos para crear un parche; para ello creamos y guardamos un archivo .wpp, y posteriormente




**Menú para la aplicación de un parche.**

Una sabia utilización del programa ahorrará tiempo y dinero en la distribución de programas

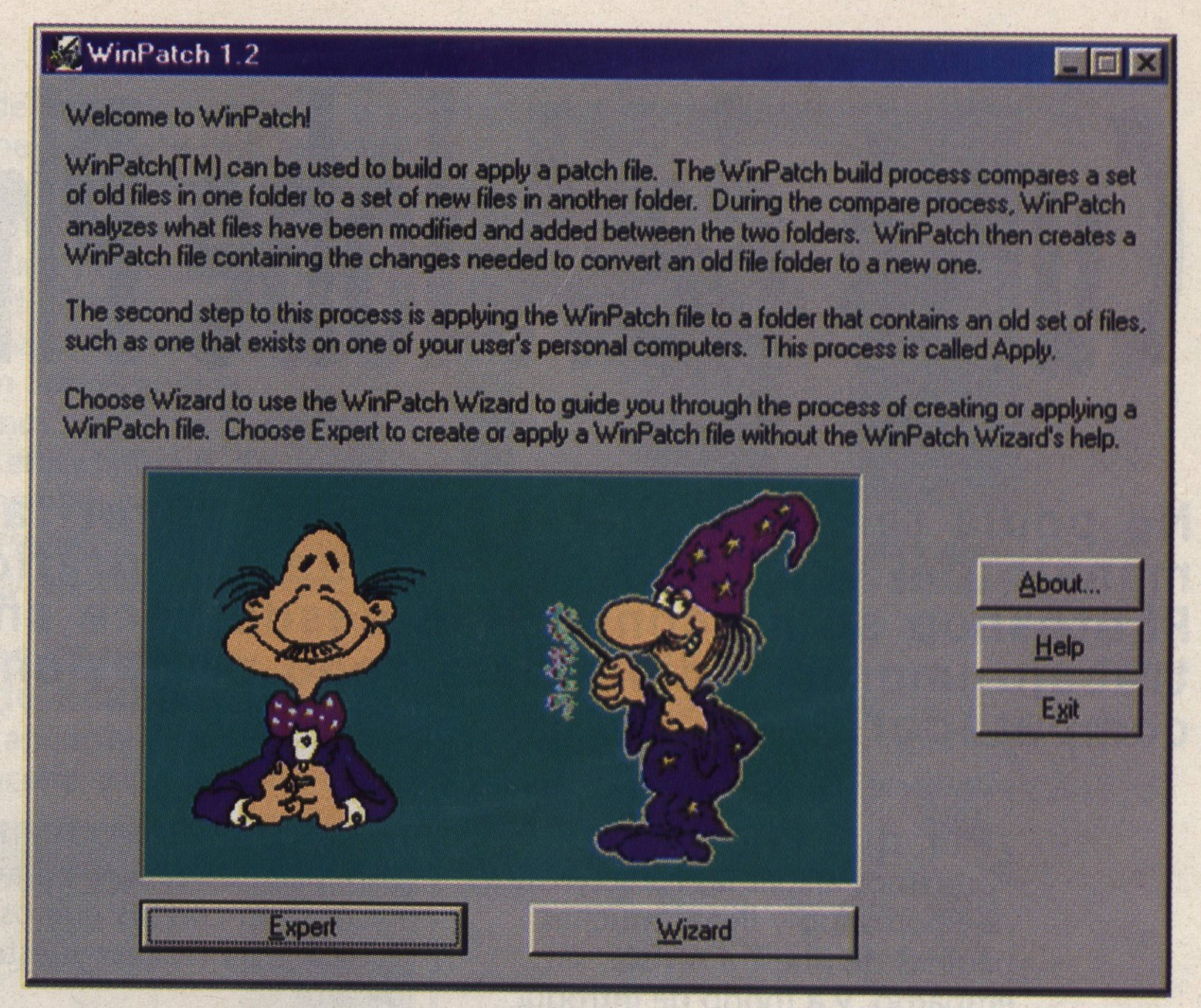
creamos el parche. Sin embargo, después de publicarlo realizamos una nueva versión, para la cual no hará falta crear un nuevo proyecto ya que ya ha sido creado, sin embargo sí será necesario crear un nuevo parche.

## Creación de un proyecto

Para crear un proyecto hay que indicarle al programa dónde están los ficheros antiguos, así como dónde están los nuevos, y en qué directorio se aplicará el parche por defecto.

- *Remove files*: puede ocurrir que en la nueva versión del programa, algunos ficheros queden obsoletos y no se utilicen para nada. *Remove Files* indica a WinPatch cuales son estos ficheros.
- *Exclude:* puede ocurrir que en el directorio donde se encuentran los nuevos ficheros haya algunos que no sean necesarios para el parche, si los incluimos en *Exclude,* éstos no serán procesados. Por ejemplo, si existe un archivo en la versión nueva que no existía en la antigua, este archivo será creado al aplicar el parche, a no ser que se le excluya.
- *Split:* con esta opción se nos permite partir el archivo del parche (usado para guardarlo en discos removibles, si éste es muy grande)
- *Create self-extracting patch program*: indicando esta opción se nos creará, como anteriormente se ha dicho, un archivo autoextraible del parche. Normalmente será la opción a utilizar, ya que no todo el mundo posee el programa de aplicación de parches.
- *Dialog Text:* será el texto que se mostrará en pantalla a la hora de instalar el parche.

Menú para la construcción de un parche.




**Es el momento de elegir el modo experto o *wizard*.**

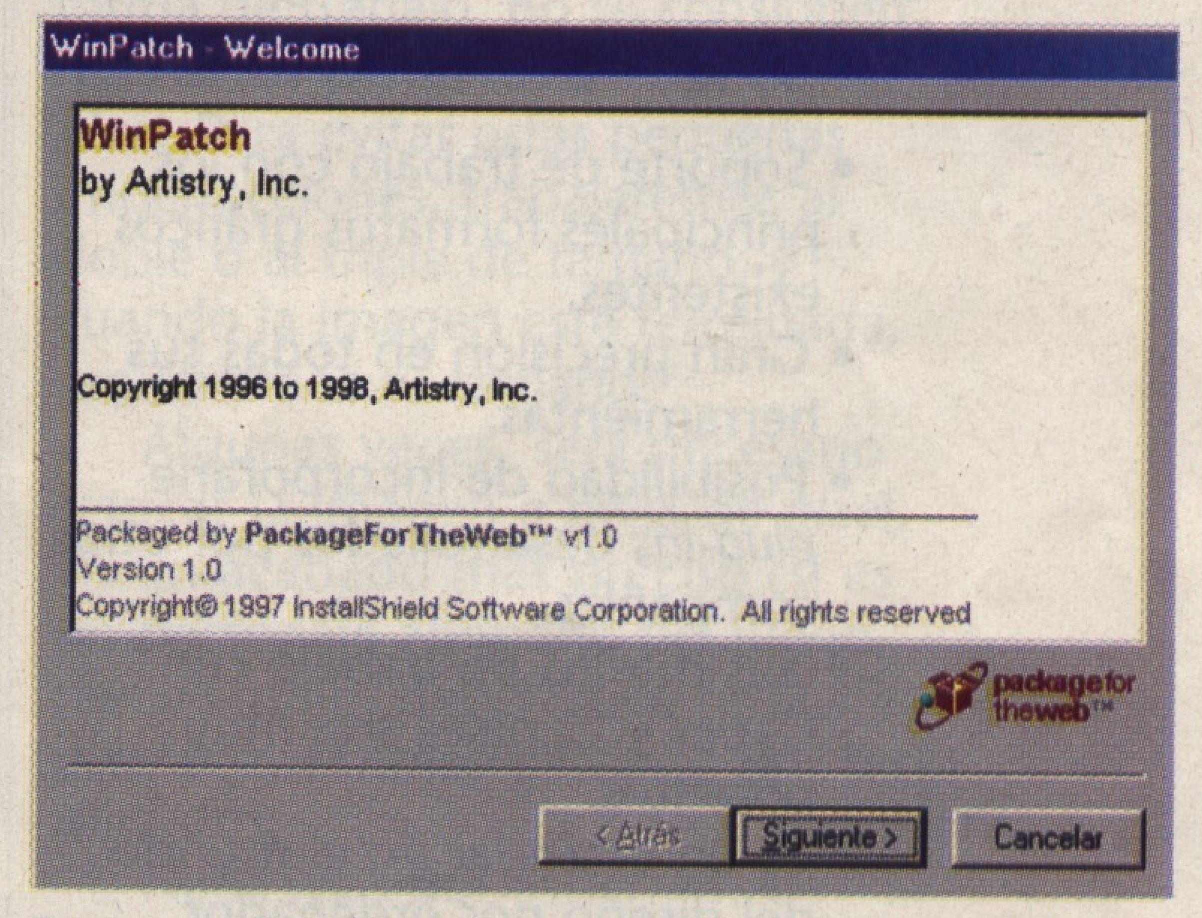
## Modo *Wizard* de WinPatch

En este modo se nos va guiando por todos los pasos necesarios para la creación o aplicación de un parche. En la primera pantalla se nos ofrecen dos posibilidades: crear o aplicar un parche. Para ello, únicamente hay que seguir las instrucciones.

## Modo *Expert* de WinPatch

En este modo se nos presentan en una misma pantalla todas las opciones anteriormente comentadas, además de la posibilidad de aplicar un parche. Una vez introducidos todos los datos podemos crear el parche, así como salvar el proyecto creado.

En este modo se nos presenta también la posibilidad de aplicar un parche: Para ello deberemos indicar el nombre de éste y el directorio a aplicarlo, así como si queremos o no copias de seguridad (*Backup copies*) de los archivos cambiados.




**Para su distribución se utiliza *Package for The Web tm*.**

Sin duda alguna, éste es el mejor programa de construcción de parches

## WPApply 1.2

Es un pequeño programa para la aplicación de parches de WinPatch, muy similar al del modo *Expert* de WinPatch.

## Para terminar

Si queréis mas información sobre WinPatch, los datos de Artistry, Inc. son:

- E-Mail: artistry@artistry.com
- Web: http://www.artistry.com
- Correo: Artistry, Inc.
  102C West Germantown Pike
  East Norriton, PA 19401
- Teléfono: (215) 499-6561

Hasta el próximo número de la revista.

**Autor: Javier Fernández**

# Curso de Adobe Photoshop (I)

**No podía faltar en vuestra revista favorita un curso para aprender el manejo del programa de retoque fotográfico por excelencia. Adobe Photoshop se ha convertido desde hace años en una herramienta de trabajo imprescindible para todo profesional del diseño gráfico por computador.**

## ¿Por qué Photoshop?

Podríamos alabar las características de Photoshop y llenar muchas páginas de nuestra revista, sin embargo, y a modo de introducción, señalaremos solo algunas de las principales:

- Soporte de trabajo con los principales formatos gráficos existentes.
- Gran precisión en todas sus herramientas.
- Posibilidad de incorporarle *plug-ins* desarrollados por otras compañías.
- Sus modos de trabajo con Capas y Canales son muy potentes.
- Es un estándar en el mundo del diseño por ordenador.

Si bien es cierto que existen multitud de herramientas de menor precio (*shareware o freeware*) que pueden competir en algunos aspectos con Photoshop, éste les supera con creces en una visión general de sus posibilidades. Ni Hamlet lo dudaría: "Photoshop, o Photoshop: ésta es la cuestión..."

## Organización de este curso

Trataremos de no cometer el típico error de aburrir a los lectores explicando únicamente las diferentes opciones y menús del programa. Iremos más allá y en cada lección explicaremos cómo abordar algún problema típico de diseño, o explicaremos algún efecto, técnica... Esto irá acompañado, a lo largo de nuestras entregas, de una exposición de todas las posibilidades de Photoshop. Así, el lector aprenderá la mayoría de conceptos practicando y sólo tendrá que aprender de una forma más «teórica» los comandos que no utilicemos directamente en nuestros ejemplos.

Intentaremos incluir todos los meses, en el CD que acompaña a la revista, una colección de *plug-ins shareware o freeware* para Photoshop con los que os podréis ayudar para realizar vuestras obras de arte.

El buzón del autor del curso (cgonmor@jet.es) queda abierto para sugerencias, peticiones y aclaraciones sobre esta serie de entregas.

## ¡Comenzamos!

Esta primera entrega va a ser eminentemente práctica. Dejaremos para la siguiente entrega el explicar las distintas herramientas de Photoshop. Nos adentramos con buen pie en nuestro curso, viendo una técnica para colorear rápidamente dibujos hechos a mano.

Todos coincidiremos en que, colorear un dibujo *pixel a pixel* supone un gasto de tiempo muy importante. Hace años, con herramientas como Deluxe Paint no había más remedio que proceder así. Incluso hoy en día, los últimos retoques de imágenes se hacen a nivel de *pixel* (al igual que muchos programadores, en última instancia, depuran su código directamente en ensamblador). Sin embargo, los resultados que podemos conseguir con Photoshop en muy poco tiempo son realmente impactantes.

Cuando dibujemos la escena en papel, al pasarla a tinta (de esta forma el escaneado será mucho mejor), debemos intentar que el dibujo quede limpio (es decir, no dar sombras con ningún tipo de trama, ni oscurecerlo).

Una vez escaneado, aumentaremos un poco el contraste de la imagen para que las líneas negras

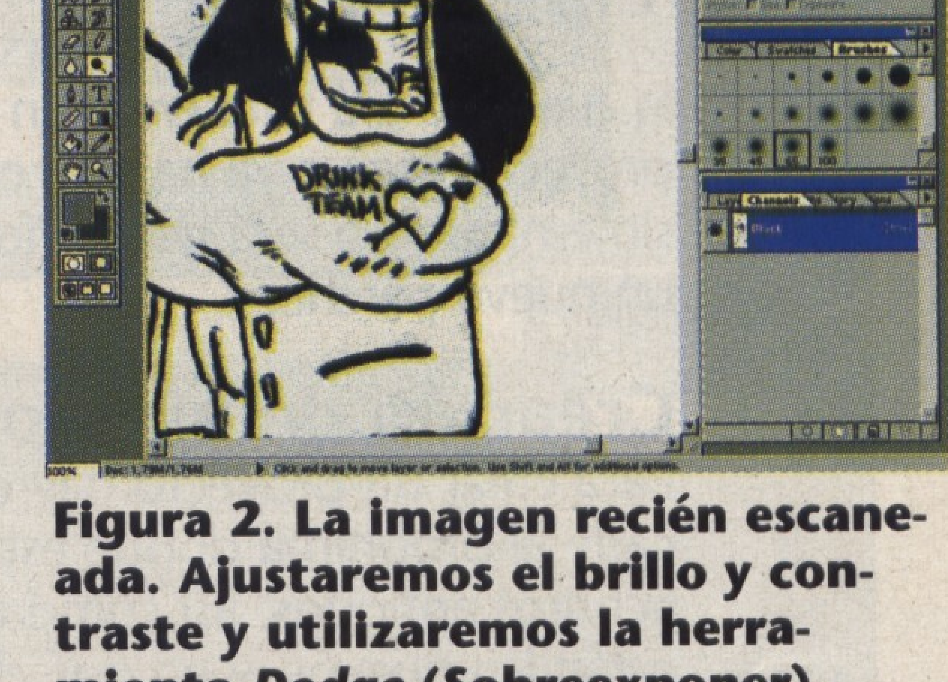


**Figura 2. La imagen recién escaneada. Ajustaremos el brillo y contraste y utilizaremos la herramienta *Dodge* (Sobreexponer).**

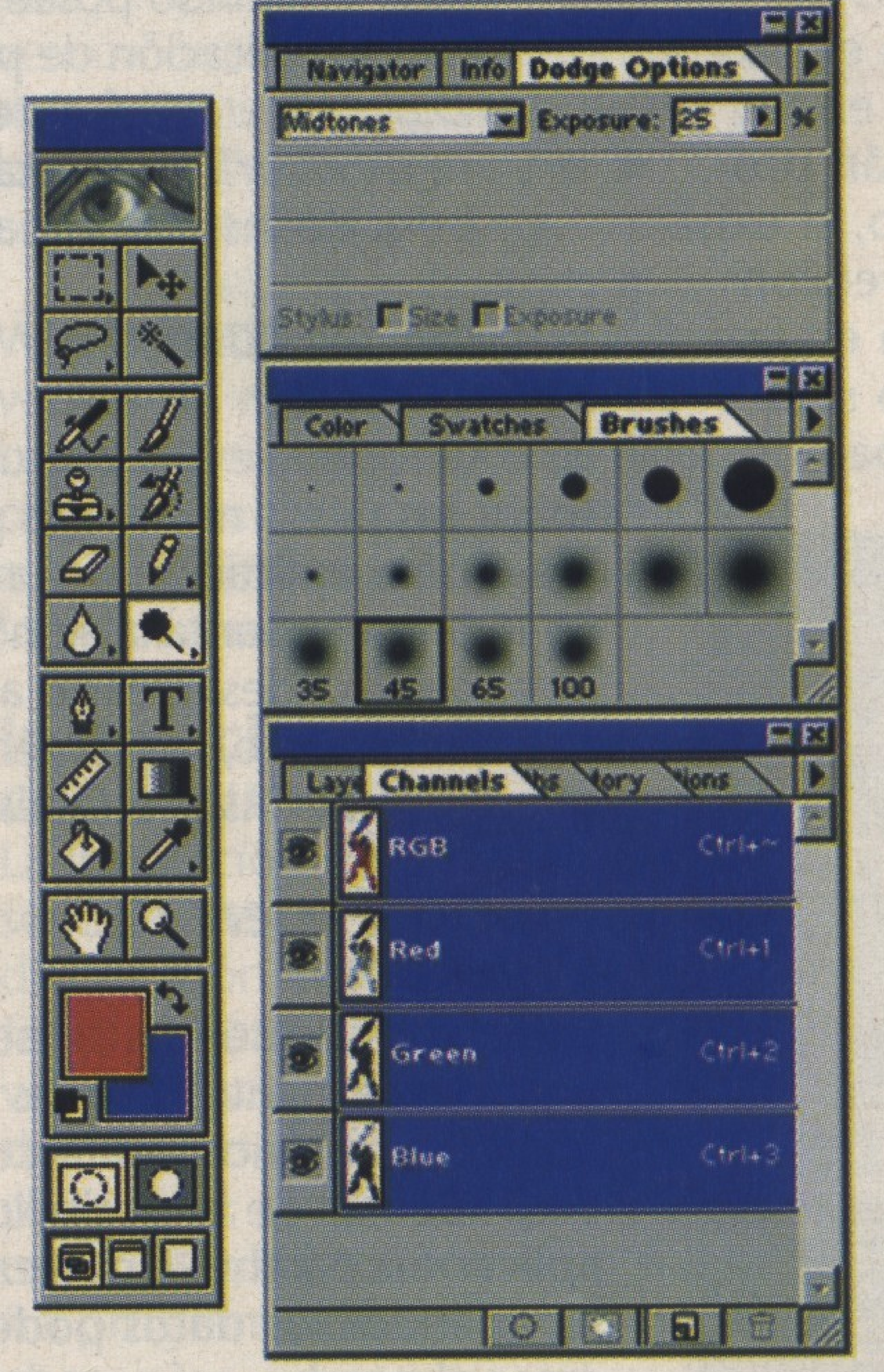



**Figura 1. El cuadro de herramientas (izquierda) y paletas (derecha) de Photoshop 5.**

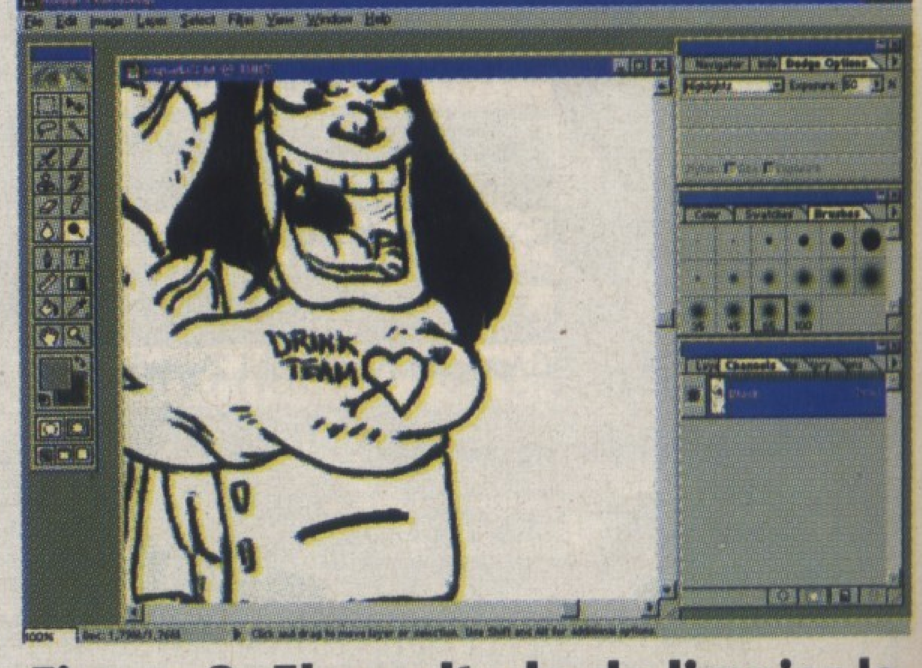


**Figura 3. El resultado de limpiar la imagen de los *pixels* grises no deseados.**


**Figura 4. Una forma de colorear directamente la imagen con la herramienta lápiz.**

se vean realmente negras (Menú: Image/Adjust/Brightness-Contrast). Aunque hemos mejorado un poco la imagen, podemos observar que todavía no está totalmente limpia (ver Figura 2). Aún quedan pixels de tonos grises que ha tomado el escáner. Para eliminarlos sin borrar las líneas que nos interesan, podemos recurrir a la herramienta de Sobreexponer *(Dodge)*, que se encuentra en el cuadro de herramientas simbolizada con una especie de lupa con la lente de color negro. Seleccionaremos esa herramienta y en sus opciones elegiremos *Highlights* (luces), y una exposición del 50%. Ahora podremos pasar sobre los *pixels* de color gris sin miedo de borrar los trazos de nuestro dibujo (ver Figura 3).

Ya tenemos el dibujo con el trazo limpio. Para dar colores planos hay varias posibilidades de trabajo. Podemos utilizar el lápiz o la herramienta de relleno. Si queremos colorear el dibujo directamente, con el lápiz y la opción *darken* (oscuro) seleccionada, dibujaremos sin miedo sobre las líneas que tenemos. Os daréis cuenta que no se borran las líneas negras que ya tenemos (Figura 4). Para conseguir una mayor precisión, podéis pulsar la tecla Bloq. Mayús. y el puntero del ratón cambiará al de precisión. En el menú de preferencias de Photoshop, en la opción de preferencias de Pantalla y Cursores, debéis activar la opción "Tamaño de pincel". Así, cuando pulséis Bloq. Mayús., el cursor cambiará a una círcunferencia que indicará los límites del pincel actual.

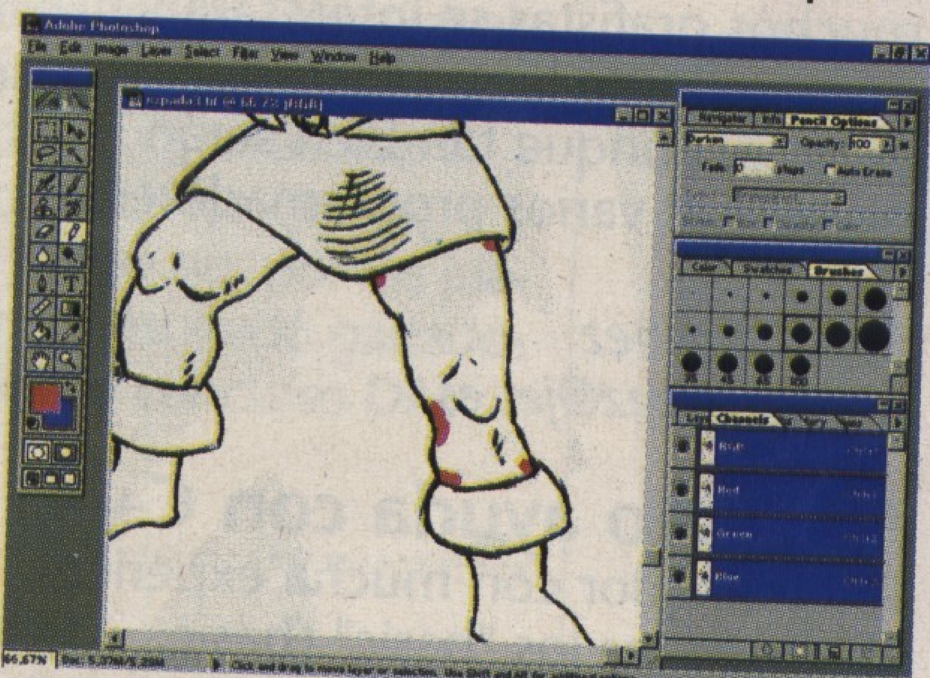
Otra forma de colorear rápidamente es cerrar la superficie a colorear con el lápiz (y la opción *darken* anteriormente seleccionada). Bastará con elegir seguidamente la herramienta de relleno y pinchar dentro de la zona cerrada. (Figuras 5 y 6).

Una vez coloreado el personaje con colores planos, vamos a darle volumen. Recurriremos a las herramientas de Sobreexponer y Subexponer (*dodge y burn* respectivamente). Ambas con la opción de tonos medios *(midtones)* activada. Haremos una pasada y pondremos las zonas de sombra con Subexponer y después daremos brillos con Sobreexponer.

Podemos ver el resultado final obtenido en la figura 8. En poco más de media hora hemos coloreado un personaje que ya está listo para incluir en nuestro videojuego.

## Algunos consejos antes de acabar...

Al escanear la imagen, deberemos hacerlo a un tamaño que


**Figura 5. Cerramos las partes donde la superficie a rellenar quedaba abierta.**

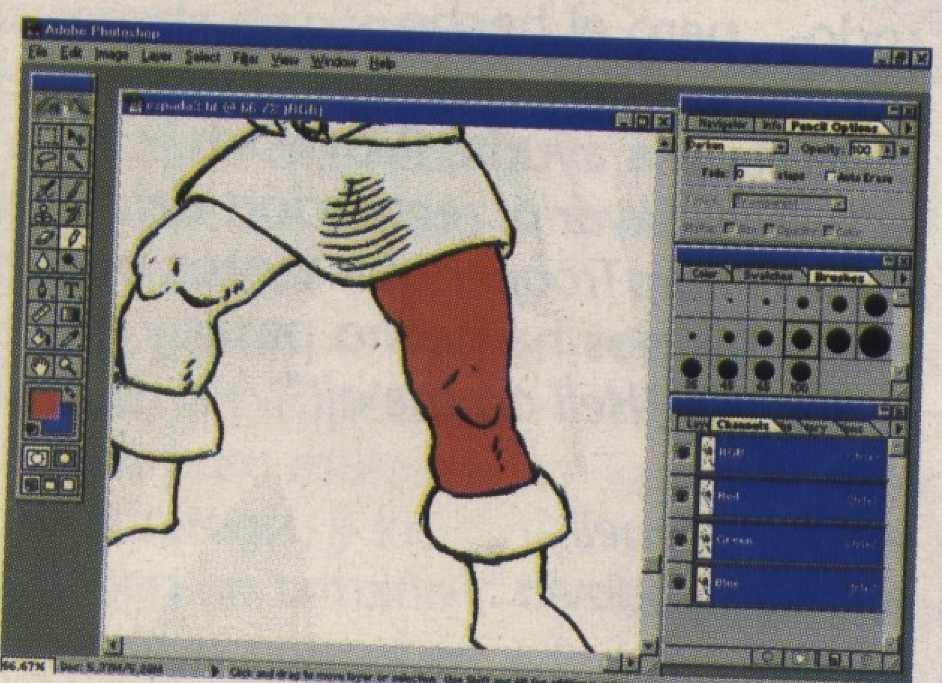

**Figura 6. Utilizamos la herramienta de relleno. De una sola pasada coloreamos toda la superficie.**


**Figura 7. El brazo de nuestro personaje está coloreado con brillos y sombras.**

sea el doble o el triple del resultado final que queremos obtener. Es decir, si el dibujo va destinado a ser de 200x600 *pixels*, el escaneado deberá hacerse de tal forma que, como mínimo la imagen original sea de 400x1200 *pixels*.

Esto es así porque, con esta forma de trabajo, no se consiguen resultados perfectos a nivel de *pixel*. Para evitar estas pequeñas imperfecciones, trabajaremos al doble o al triple de tamaño, y cuando la imagen esté terminada, reduciremos su tamaño.

Algunas veces, será necesario retocar la imagen a nivel de *pixel*, en un acabado más preciso (si es un personaje animado del juego, por ejemplo, deberemos retocarlo *pixel a pixel* antes de darlo por terminado).

Nada más por este mes. Os esperamos en la próxima entrega de este curso de Photoshop. Un saludo.

**Por Carlos González Morcillo**
**cgonmor@jet.es**



**Figura 8. Esquema de los pasos seguidos.**

# El correo del lector

## Tablón de anuncios

**Sin duda hoy por hoy Internet se ha convertido en un ilimitado foro de opinión, encuentro, contacto e información. Por eso, hemos seguido buceando en ella para buscar todo aquello que vosotros, los seguidores de DIV, deseáis "lanzar al espacio"; de modo que seguimos difundiendo los mensajes que aparecen en las páginas web.**

Estamos plenamente convencidos de que DIV es todo un éxito. Sobre todo a juzgar por la multitud de páginas web dedicadas a este programa, así como las diversas iniciativas surgidas en torno a él. Por eso hemos decidido seguir dedicando este correo del lector a servir como una especie de tablón de anuncios.

### Buscan gente para formar un grupo de desarrollo

Ya somos unos cuantos en el grupo y hemos empezado un desarrollo pero todos los que estéis interesados mandad un e-mail con la referencia Grupo-Div a la dirección indicada, diciendo qué tipo de tarea podéis realizar, gráficos, sonido, programación,etc...

Entre todos afrontaremos los distintos proyectos u otros que fueran posteriormente propuestos y elegidos por el grupo. Seria muy interesante que varios grafistas, muchos grafistas se interesaran, porque desde luego es lo que más escasea, aunque tambien serán necesarios varios programadores.

*Patxi Sánchez*
*patxisanchez@jet.es*

### Necesito ayuda con C++

Programador con mucha experiencia en todos los "basic" (basic, gwbasic, qbasic, visual basic), y C, busca ayuda con C++, ya que no lo domina del todo... pero el hecho de no tener conexión a Internet dificulta esa ayuda. Proyectos actuales: *Skydrip,* recreación del clásico *ParaTrooper* que tantos buenos ratos nos ha hecho pasar y *AirHunt2: "Hell on the air".*

*Arturo Granell*
*sinmail@todavia :)*

### Busco equipo de programación

Soy Grafista 3d 8 (con mucha experiencia en diseño con 3dStudio) y Beta Tester (MUY crítico). No ha hecho nada con DIV pero quiere crear el MEJOR juego tipo *Quake* y, también, programar el MEJOR simulador de carreras de coches.

*David Talavera*
*mie@databasedm.es*

### Busco colaboradores... y novia

Programador y dibujante/grafista 2D está buscando ayuda para su proyecto de creación de una aventura gráfica. Necesita: programadores auxiliares, grafistas, músicos, guionistas, repartidores de pizza y novias.

*Carlos Asensio*
*proto@mixmail.es (preferente)*
*cam2@alu.ua.es*
*www.multired.com/computa/caasemar*

### Se ofrece...

Programador con domino de casi todos los comandos del div, con experiencia con lenguajes antiguos (basic, qbasic, clipper/dbase...), grandes ideas y unos cuantos proyectos creados y otros tantos por desarrollar, se ofrece para trabajar, siempre y cuando le interese el juego.

*Fermín Vicente*
*ferminho@retemail.es*

### Ayuda para DIV

Programadora de alto nivel (es decir, programadora de engines completos) y dibujante/grafista 2D, busca ayuda con Div.

*Belén Albeza*
*benko@olemail.com (preferente)*
*http://pagina.de/benko*



### Fondo Documental sobre DIV

Estoy creando un fondo documental sobre DIV (algo así como un *knowledge base*) para el que necesito vuestra ayuda. Tenéis que escribir documentos de texto plano (TXT) sobre las cosas que se os ocurran. Rutinas para realizar más fácilmente determinadas tareas. Documentos sobre Inteligencia Artificial (genérica o aplicada a DIV). Programación de DLL's. Trucos o métodos gráficos, etc... Todos los documentos aparecerán firmados por su autor y la forma de acceder a ellos será a través de un cuadro de búsqueda por palabras clave. La información aparecerá en la página:

*http://www.fortunecity.com/skyscraper/liz/356. Gracias por vuestra ayuda. jota@eucmos.sim.ucm.es*

Redacción DIV Manía

### Esperamos vuestros comentarios

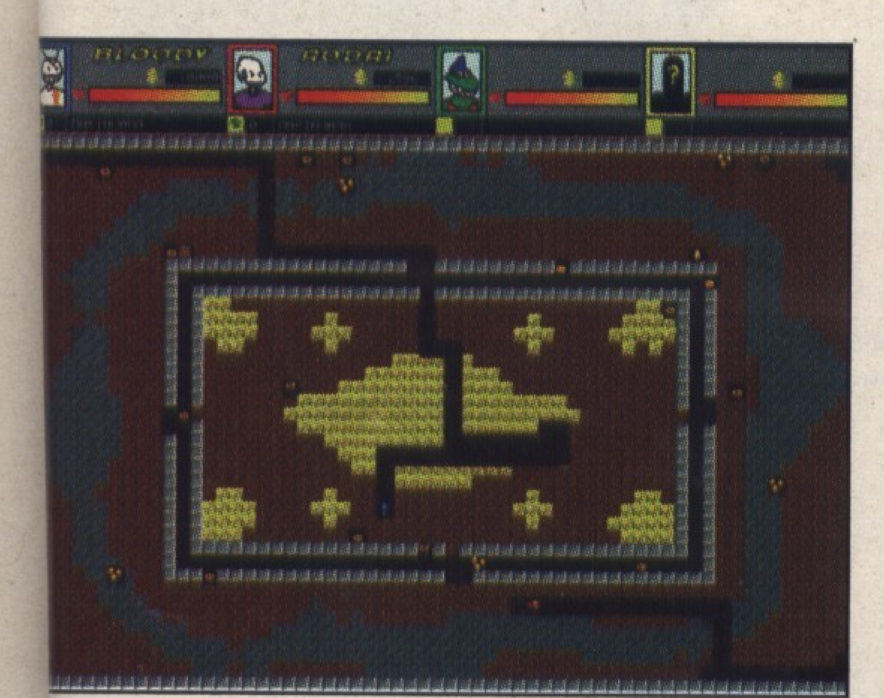
Agradecemos los comentarios que nuestros lectores nos han hecho en sus cartas. Los tendremos en cuenta. Os recordamos nuestra dirección postal y de Internet.

Revista Div Manía
Sección Correo
C/ Alfonso Gómez 42, nave 1-1-2
28037, Madrid

**divmania@prensatecnica.com**

# Contenido CD-Rom

Nuestro CD trae esta vez propuestas más que interesantes. Desde una demo de Tokenkai, la demostración de las posibilidades de DIV a nivel profesional, hasta una buena pléyade de programas shareware para todos los gustos, pasando, como no, por los juegos que han resultado ganadores de nuestro concurso DIV.





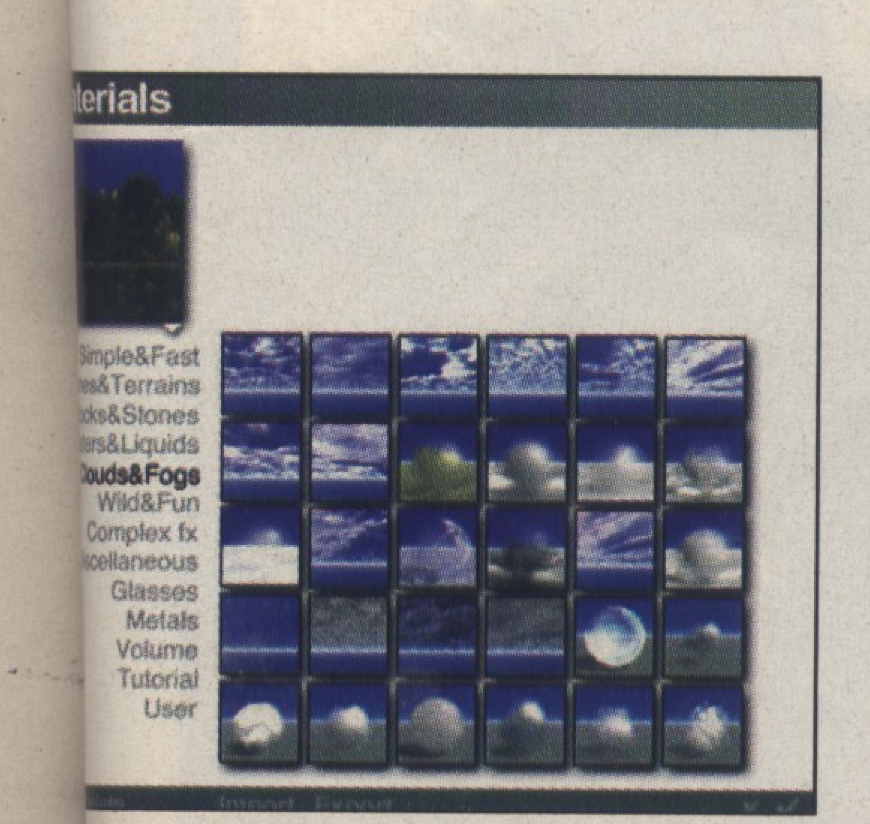

## DEMOS

### Tokenkai

Es la demostración "viviente" de que DIV es capaz de hacer cosas muy buenas. Se trata de un curiosa mezcla de géneros (arcade y estrategia) que recuerda a un título que hizo las delicias de muchos jugadores de todo el mundo: *Command & Conquer.* Como en aquél, la perspectiva es aérea, al más puro estilo de un juego de estrategia, y los personajes tienen un tamaño más pequeño de lo normal en un arcade. Nuestra misión es acabar con el imperio de la droga que domina las ciudades en un futuro no muy lejano. Nuestro protagonista, una especie de hermano menor de Rambo armado hasta los dientes (de valor y de diversos artefactos explosivos). Ahora puedes probarlo con esta demo.

### Bryce 3D

Nuestro CD incluye la demo de este programa. Aunque en nuestra revista os hablamos de la versión 4, os ofrecemos la versión anterior para que os hagáis una idea de las posibilidades de este software generador de entornos tridimensionales que destaca por su gran potencia, sus amplias posibilidades y su sencillez de manejo.



## JUEGOS COMPLETOS

### Disco Fighter

Os ofrecemos el juego que se ha alzado con la victoria en el concurso de creación de juegos con DIV. Se trata de un divertido juego que sigue el estilo del archifamoso *Street Fighter.* Como en aquel, *Disco Fighter* nos va a dar pie a desfogarnos pegando patadas y puñetazos a diestro y siniestro.

### PA-PÚ

El subcampeón de nuestro concurso es un programa que versiona un clásico de los videojuegos (adivinad cúal). El juego destaca por su divertido sonido, sus simpáticos gráficos y, especialmente, su jugabilidad, que le convierte en muy adictivo. Disfrutadlo.

### Bloody's War '99

La medalla de bronce es para este juego pensado exclusivamente para multijugador. Permite que se enfrenten hasta cuatro jugadores. La paciencia debe ser el arma fundamental a la hora de enfrentarse a un juego que plantea ir recogiendo objetos del suelo, por el que nos moveremos gracias a la acción de un taladro.



CURSOS DE JUEGOS

Nuestro CD también incluye los pequeños programas que acompañan a las diversas secciones de nuestra revista: aventura, rol, DIV, etc.

SHAREWARE

## ACDSee 2.41
Uno de los más útiles visores en su última versión shareware.

## CD Box Labeler 1.1
Te permite crear etiquetas y libretos para CDs. Podrás decidir con total libertad el contenido de la carátula o etiqueta de tu CD.

## Copernic 99 301
Este agente de software gratuito encontrará con exactitud lo que estés buscando en Internet.



## Cyber-Info Webmail
Un sencillo programa que te avisa cuando tu cuenta de HotMail tiene nuevos mensajes. Hotmail Notify trabaja desde la bandeja de sistema, notificando visualmente o mediante un aviso sonoro que tienes nuevos mensajes esperando.

## Download Accelerator 3.5
Download Accelerator acelera la recepción de ficheros fragmentando los mismos a partir de un tamaño definido por nosotros mismos.

## Eudora Pro 4.2
El cliente de correo electrónico más popular para Windows 95/NT.

## FTP SERV-U 2.50 Build 4
Un excelente lector y buscador dentro de *newsgroups*. A la hora de buscar podrás especificar intereses diferentes, cada uno con su propio criterio, y escogiendo los *newsgroups* donde buscar.

## Hyper Maker
Aplicación que te permitirá transformar páginas Web en publicaciones comprimidas y encriptadas, para su distribución. Incluye un visualizador libre de *royalties*.

## ICQ Plus 2.01
Cuando instalas ICQ Plus, una nueva opción aparece en el menú principal. Desde este menú puedes cambiar el fondo de las imágenes usada para los cuadros de diálogo, botones, controles...

## mIRC 2.01
Es uno de los mejores clientes de IRC, además de ser todo un clásico. Si quieres charlar con gente de todo el mundo a través de Internet, no busques más.

## Nendo 1.1
Nendo es una potente herramienta de modelaje en 3D que, sin embargo, destaca por su facilidad de uso.

## NetInfo 3.4
Programa multitarea que proporciona herramientas para recopilar información sobre la conexión actual de un ordenador a Internet.

## News Rover 4.42
Útil lector de *news,* gracias al cual podremos organizarlas de forma cómoda y acceder a ellas problema.

## Organizador de Música
Gracias a él podrás ordenar las bandas de música si tienes tu aparato conectado al equipo, así como las melodías que poseas MP3.

## PACT JumpReg 99.6
JumpReg te puede llevar hasta las palabras clave de más uso del Registro, ahorrando tiempo y esfuerzo.

## PPPshar Pro 1.51
Con PPPshar, podrás conectar todos los ordenadores de tu red a Internet usando una sóla conexión telefónica.

## SmartFTP 1.0 Build
SmartFTP nos proporciona un interfaz de múltiples ventanas con soporte para arrastrar y soltar, permite reanudar transferencias erroneas, incluyendo envíos, descargas y transferencias remotas de servidor a servidor (FXP).

## SynEdit 0.40 beta
SynEdit es un excelente editor de textos orientado a programadores. Una herramienta de programación que facilitará la tediosa tarea de programar.

## UltraEdit Professional
Es más que un reemplazo del Bloc de Notas, es un editor de textos completísimo.

## URLSenty 1.14
Escanea de forma instantánea tus páginas Web favoritas (o las paginas Web que quieras) en busca de cambios o actualizaciones.

## Viruscan 4.0.3
Uno de los más importantes antivirus del mercado nos llega en su última versión shareware.

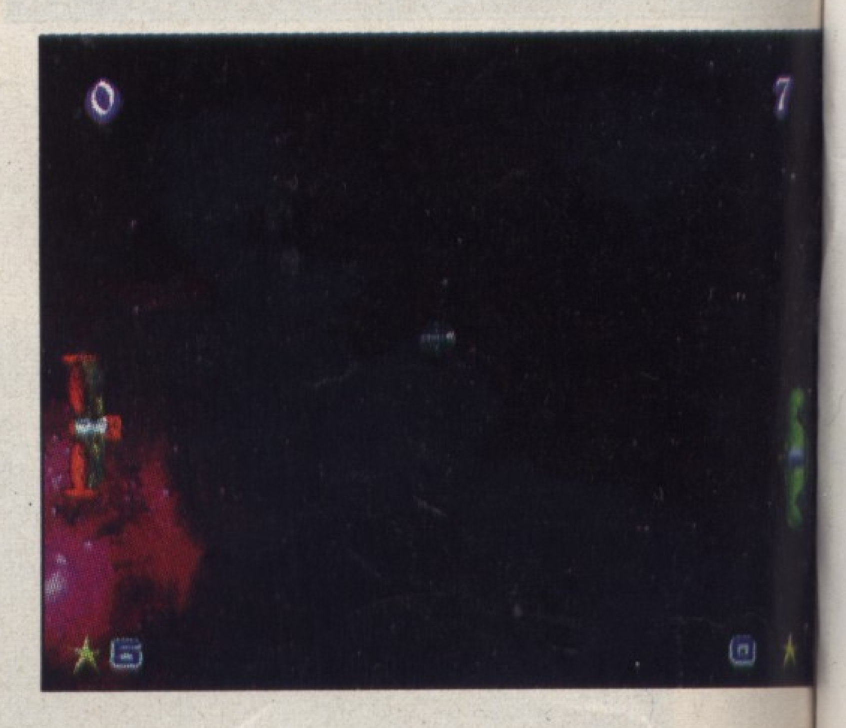
## WebAurora 98 1.01
El editor Web para Windows 95 y 98. Incorpora cinco asistentes para la creación de documentos y cuatro editores.

## WinAmp 2.22
Con este programa podremos escuchar música con la más alta fidelidad.

## Winzip70SR-1
El más importante compresor del mercado. Gracias a él podremos descomprimir gran cantidad de programas de juegos, incluidas algunas de las demos que os ofrecemos. Una herramienta imprescindible.

## WorkTime 1.1
Con Worktime puedes controlar cuánto tiempo pasas delante del ordenador trabajando, ya que te avisa cuando tu jornada ha terminado. Cuenta con dos tipos de alarma (normal y de cuenta atrás), con soporte multiusuario y con un excelente generador de estadísticas.





# CONTENIDO DEL CD ROM




**DEMO TOKENKAI**
Un juego que demuestra lo que se puede llegar a hacer con DIV. Tokenkai nos sitúa en una ciudad futurista donde el imperio de la droga se ha hecho con el control. Nuestra misión: acabar con los narcos, naturalmente. El juego es una interesante mezcla de dos populares géneros: arcade y estrategia. Como en éste último, la perspectiva es aérea; como en aquél, hay que tirar de metralleta si queremos ser alguien y cumplir la misión que nos han encomendado.

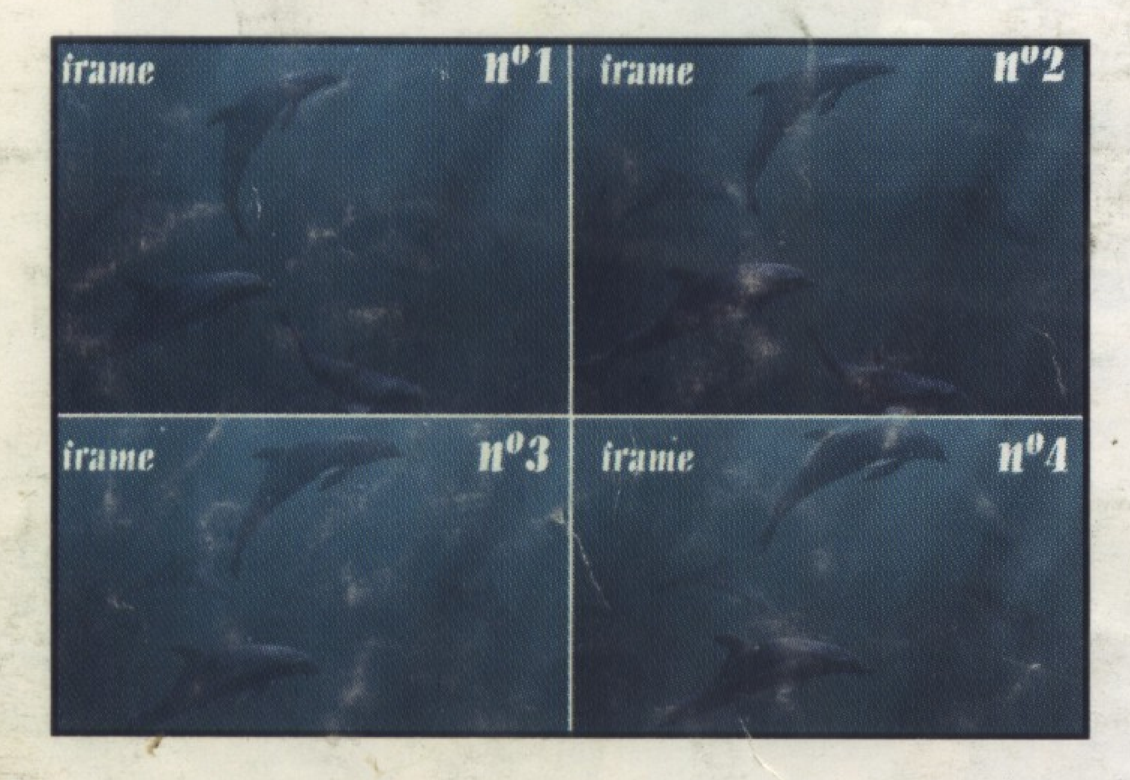
**BRYCE 3D**
Incluimos una demo de este programa. Se trata de una versión anterior a la que os comentamos en nuestras páginas, pero nos ha parecido adecuado incluirla para que os hagáis una idea de sus posibilidades. Se trata de un software generador de entornos tridimensionales que destaca por sus amplias posibilidades, gran potencia y sencillez de uso.

**JUEGOS GANADORES**
Por supuesto, incluimos los juegos ganadores de nuestro concurso de programación para que les echéis un vistazo, os divirtáis con ellos y, si es posible, os animéis a participar.

**CURSOS DE JUEGOS**
Igualmente, en nuestro CD podéis encontrar los pequeños programas que acompañan a las diversas secciones de nuestra revista.

**SHAREWARE**
Pero además, incluimos una larga lista de programas shareware que a buen seguro os resultarán de gran utilidad:
ACDSee 2.41, CD Box Labeler 1.1, Copernic 99 301, Download Accelerator 3.5, Eudora Pro 4.2, FTP SERV-U 2.5ª Build 4, Hyper Maker, ICQ Plus 2.01, mIRC 2.01, Nendo 1.1 NetInfo 3.4, News Rover 4.42, Organizador de Música PACT JumpReg 99.6, PPPshar Pro 1.51, SmartFTP 1.0 Build, SynEdit 0.40 beta, UltraEdit Professional, URLSenty 1.14, Viruscan 4.0.3, WebAurora 98 1.01, WinAmp 2.22, Winzip70SR-1 y WorkTime 1.1.

REPORTAJE Nos adentramos en todo lo que un programador necesita en su trabajo.



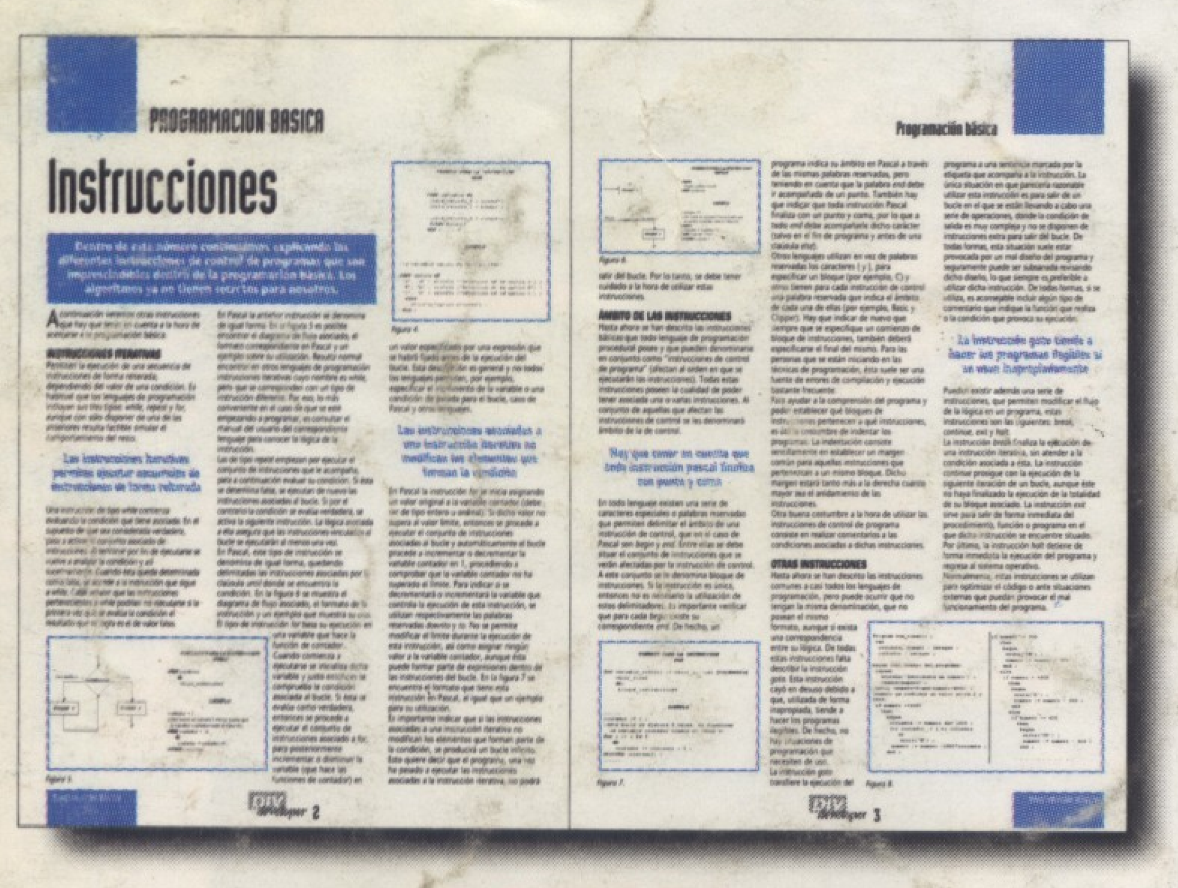

LIBRERÍAS Sonidos, fuentes, texto... Las mejores librerías a vuestro servicio.



PROGRAMA DEL LECTOR Conoce a los ganadores de nuestro concurso del mes.




# DIV manía CON EL MEJOR CONTENIDO




ACTUAL




EXHAUSTIVO




DIDÁCTICO

# Y MUCHO MÁS...